Premium

TSX ETC 100 EtherNet/IP Communication Module
User Manual

7/2012

31008211.03

www.schneider-electric.com

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information that is contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

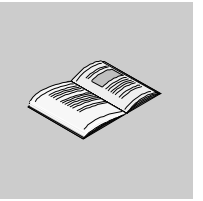
All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

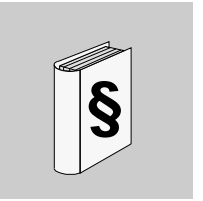
Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

Table of Contents



Safety Information



Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a Danger safety label indicates that an electrical hazard exists, which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER

DANGER indicates an imminently hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠ WARNING

WARNING indicates a potentially hazardous situation which, if not avoided, **can result in** death or serious injury.

⚠ CAUTION

CAUTION indicates a potentially hazardous situation which, if not avoided, **can result in** minor or moderate injury.

NOTICE

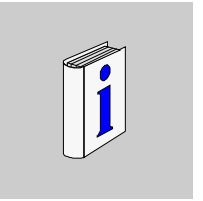
NOTICE is used to address practices not related to physical injury.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

About the Book



At a Glance

Document Scope

This manual describes the use of the Premium TSX ETC 100 EtherNet/IP communication module. This manual presents a continuing sample configuration. The features of the module are described as they are encountered in the course of this continuing sample configuration.

The specific configuration settings contained in this manual are intended to be used for instructional purposes only. The settings required for your specific EtherNet/IP configuration may—and probably will—differ from the examples presented in this manual.

Validity Note

This document is valid from Unity Pro 6.0.

The technical characteristics of the devices described in this manual also appear online. To access this information online:

Step	Action
1	Go to the Schneider Electric home page *www.schneider-electric.com*.
2	In the **Search** box type the reference of a product or the name of a product range. ● Do not include blank spaces in the model number/product range. ● To get information on a grouping similar modules, use asterisks (*).
3	If you entered a reference, go to the **Product datasheets** search results and click on the reference that interests you. If you entered the name of a product range, go to the **Product Ranges** search results and click on the product range that interests you.
4	If more than one reference appears in the **Products** search results, click on the reference that interests you.

Step	Action
5	Depending on the size of your screen, you maybe need to scroll down to see the data sheet.
6	To save or print a data sheet as a .pdf file, click **Download XXX product datasheet**.

The characteristics that are presented in this manual should be the same as those characteristics that appear online. In line with our policy of constant improvement, we may revise content over time to improve clarity and accuracy. If you see a difference between the manual and online information, use the online information as your reference.

Related Documents

For additional information, you can also refer to the online help files for both the:

- Unity Pro software
- Unity Pro EtherNet/IP Configuration Tool software

Title of Documentation	Reference Number
Advantys STB EtherNet/IP Network Interface Applications Guide	31008204

You can download these technical publications and other technical information from our website at www.schneider-electric.com.

User Comments

We welcome your comments about this document. You can reach us by e-mail at techcomm@schneider-electric.com.

Installation

1

Overview

This chapter provides installation information for the TSX ETC 100 EtherNet/IP communication module.

What Is in This Chapter?

This chapter contains the following topics:

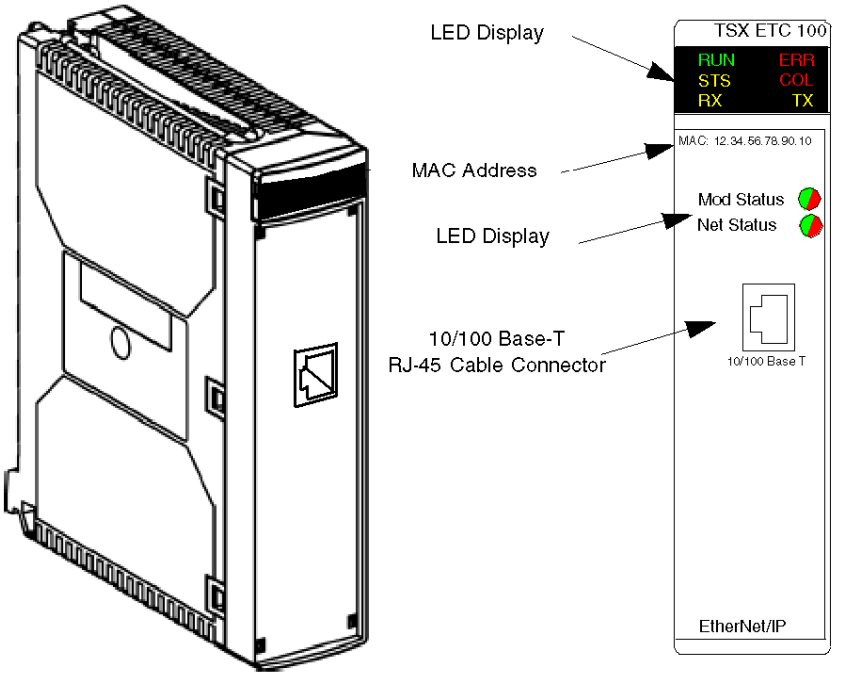
Hardware Installation

Overview

The following information describes how to install the TSX ETC 100 EtherNet/IP communication module.

External Features

Use the following illustration to identify the external features of the TSX ETC 100 module.



LEDs

The TSX ETC 100 EtherNet/IP communication module presents the following LED indicators:

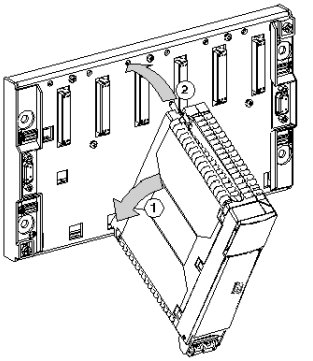
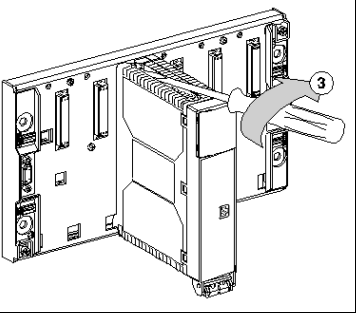
- Running
- Error
- Status
- Collision
- Reception Activity (RX)
- Transmission Activity (TX)

- Module Status
- Network Status

For a description of these LEDs, and how to use them to diagnose the communication module, refer to the topic LED Indicators for the TSX ETC 100 Module *(see page 198)*.

Mounting the Module

The TSX ETC 100 is mounted in the rack slot of a Premium/Atrium PLC station. It can be installed in any available slot (except in the offset X Bus racks). To mount the TSX ETC 100 module:

Step	Action	Illustration
1	Place the prongs at the lower back of the module into the centering holes on the lower part of the rack.	
2	Swivel the module up and back to bring it into contact with the rack and the pin connectors.	
3	Secure the module to the rack by tightening the screw on the upper part of the module. **Note:** Maximum tightening torque is 2.0. N.m.	

Wiring the Ethernet Connector

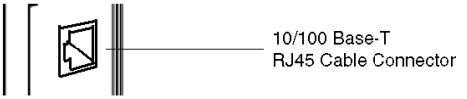
⚠ WARNING

HAZARD OF ELECTRICAL SHOCK OR BURN

Connect the ground wire to the protective earth (PE) terminal before you establish any further connections. When you remove connections, disconnect the ground wire last. The Ethernet cable shield must be connected to PE ground at the Ethernet switch.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

The TSX ETC 100 module communicates over an EtherNet/IP network through a single RJ45 connector located in the upper half of the module.



Module Specifications

Related Documentation

Refer to the *Ethernet Communication Hardware Installation* chapter in the *Premium and Atrium Using Unity Pro Ethernet Network Modules User Guide* for more detailed information on the installation.

Specifications

Specifications	
Communication Port	One auto-negotiating 10/100Base-T shielded twisted pair (RJ-45 connector) port.
Current Consumption	490 mA (running) 420 mA (stopped)
Power Dissipation	2.5 W
Fuse	None
Operating Conditions	
Temperature	0...+60° C
Humidity	0...95% Rh non-condensing @ 60° C
Altitude	2000 m (6561.68 ft) operating 3000 m (9842.52 ft) transport
Vibration	5...8.4 Hz @ 14 mm d.a.
	8.4...150 Hz @ 2 g
Storage Conditions	
Temperature	-40...+85° C
Humidity	0...95% Rh non condensing @ 60° C
Free Fall	1 m unpackaged
Shock	3 shocks / axis, 15 g, 11 ms

Software Compatibility

The TSX ETC 100 is compatible with Unity Pro XL programming software version 4.0 and higher.

Standards

The TSX ETC 100 module complies with the following standards:

- UL 508
- CSA 22.2-142
- CE
- C-TICK
- ODVA

Communication Modules per Station

The maximum number of communication modules, including but not limited to the TSX ETC 100 EtherNet/IP communication module, that can be installed in a single station—including an extended station—is determined by the CPU serving that station:

CPU	Maximum Number of Communication Modules per Station
TSX H57 24	2
TSX H57 44	4
TSX P57 104	1
TSX P57 154	1
TSX P57 204	2
TSX P57 0244	1
TSX P57 254	2
TSX P57 304	3
TSX P57 354	3
TSX P57 454	4
TSX P57 554	4
TSX P57 1634	0
TSX P57 2634	1
TSX P57 3634	2
TSX P57 4634	3
TSX P57 5634	3
TSX P57 6634	3

Configuring the TSX ETC 100 EtheNet/IP Communication Module

2

Overview

This chapter shows you how to use Unity Pro programming software and the Unity Pro EtherNet/IP configuration tool to select and configure the TSX ETC 100 EtherNet/IP communication module.

NOTE: The instructions presented in this chapter include specific choices made for a sample project. Your Unity Pro project may include different choices that are appropriate for your specific configuration.

What Is in This Chapter?

This chapter contains the following sections:

2.1 Creating a Project in Unity Pro

Overview

This section provides information about:

- selecting Premium modules in Unity Pro
- launching the Unity Pro EtherNet/IP configuration tool

NOTE: For detailed information about how to use Unity Pro, refer to the online help and documentation DVD that come with the Unity Pro XL programming software.

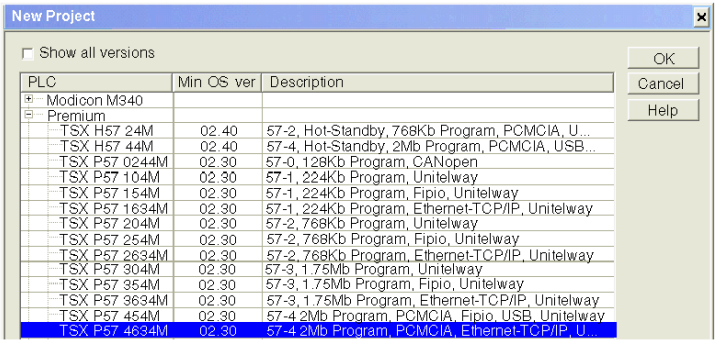
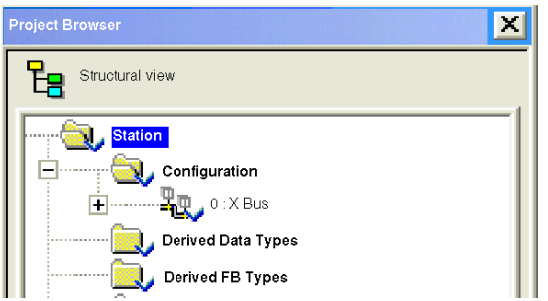
What Is in This Section?

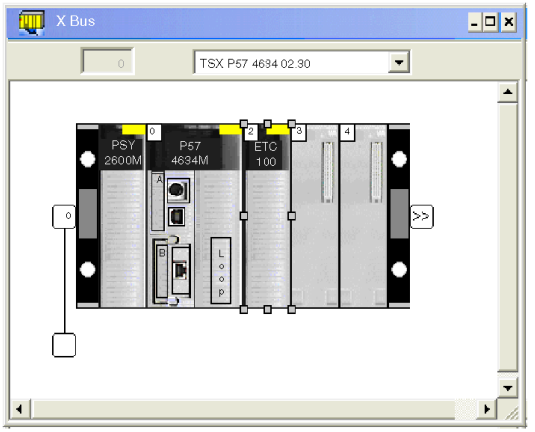
This section contains the following topics:

Creating a Project

Using Unity Pro

Use Unity Pro to create a new project. The following steps provide an example of how to create a project using Unity Pro:

Step	Action
1	Launch Unity Pro.
2	In the Unity Pro main menu, select **File** →**New...**. The **New Project** window opens displaying a list of Schneider-Electric controller types.
3	In the **New Project** window, open the Premium sub-list and select a controller. In this example, the TSX P57 4634M controller is selected: 
4	Click **OK**. The **Project Browser** opens: 
5	In the **Project Browser**, double click **Local Bus**. Unity Pro displays: • the **Hardware catalog**, and • a **Local Bus** window with the selected CPU in the second position (slot 0) and a TSX PSY 2600M power supply in the first position

Step	Action
6	In the **Hardware catalog**, use your mouse to drag a TSX ETC 100 EtherNet/IP communication module from the **Communication** section to a position in the backplane. In this example, the module is placed in the third position (slot 2).
7	The modules that you have selected are now displayed in the backplane. 
8	To open the configuration window for the TSX ETC 100, do one of the following: ● double click the left mouse button on the TSX ETC 100 module in the **Local Bus** window above, or ● click the right mouse button on the module, then select **Open Module...** in the popup menu The module configuration window opens, where you can configure the properties for the TSX ETC 100.

Configuring the TSX ETC 100 EtherNet/IP Communication Module

Overview

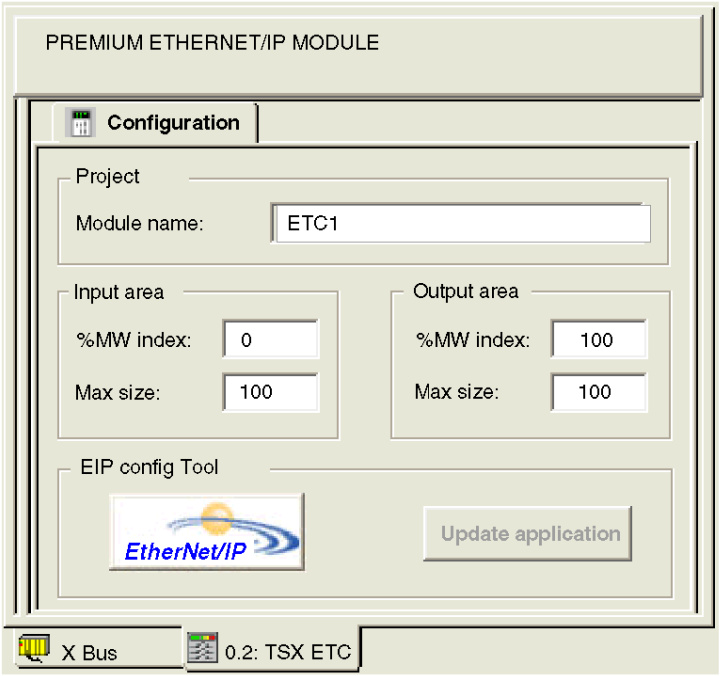
To configure properties for the TSX ETC 100, you need to:

- complete the **Configuration** page of the module properties window
- launch the Unity Pro EtherNet/IP configuration tool, where you can complete the process of editing properties for EtherNet/IP modules and devices
- add the completed EtherNet/IP module and device edits to the Unity Pro project configuration in the form of derived data types

The following steps present an example of how to configure the communication module. Your own configuration may differ.

Setting Input and Output Memory Addresses and Naming the Module

The **Configuration** page looks like this:



In the **Configuration** page, perform the following steps to name the module, and to set addresses and sizes for both inputs and outputs:

Step	Action
1	In the **Project** section, type in a name for your module in the **Module name** input box - in this example: **ETC1** **Note:** After the module name is entered and the EtherNet/IP configuration is validated (by clicking the button), the module name cannot be edited.
2	In the **Input area** and **Output area**, type in the size and starting position of both the inputs and outputs. These values can later be edited. For this example, the following values are entered: In the **Input area**: • In the **%MW index** field, type in a starting address for inputs—in this example: **0**. • In the **Max size** field, type in the maximum number of 16-bit words dedicated to inputs—in this example: **100**.) In the **Output area**: • In the **%MW index** field, type in a starting address for outputs—in this example: **100**. • In the **Max size** field, type in the maximum number of 16-bit words dedicated to outputs—in this example: **100**.) **Notes**: • The inputs and outputs can be located at any available address, and do not need to be located in adjacent areas. It is important only that the space allocated to inputs and outputs do not overlap • The specified %MW range for both inputs and outputs must be available in the CPU. For more information, refer to the Unity Pro help file topic *Processor Configuration Screen*.
3	In Unity Pro, select **Edit** →**Validate** (or click the **Validate** button) to: • save the EtherNet/IP module name—which becomes a non-editable, read-only value • save the address and size settings for inputs and outputs, and • start up the EtherNet/IP configuration tool

Launching the Ethernet/IP Configuration Tool

After you have saved both the EtherNet/IP module name and the input and output settings, launch the EtherNet/IP configuration tool by clicking on the EtherNet/IP button:

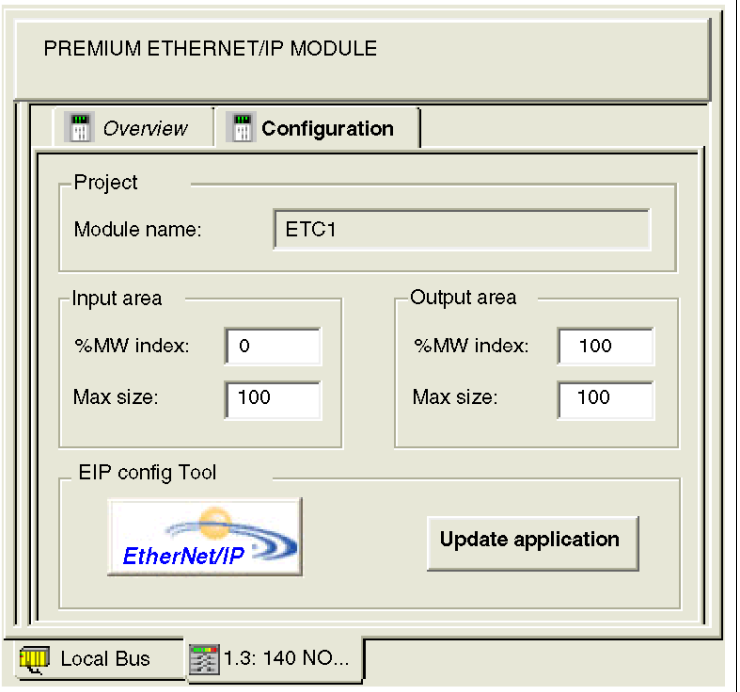
The EtherNet/IP configuration tool opens for editing. If EtherNet/IP device configurations have previously been edited and saved, they will be displayed.

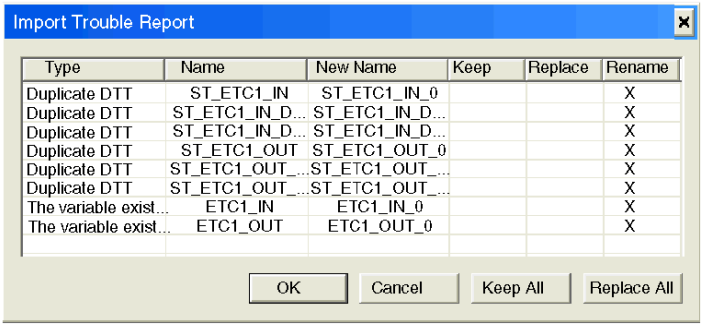
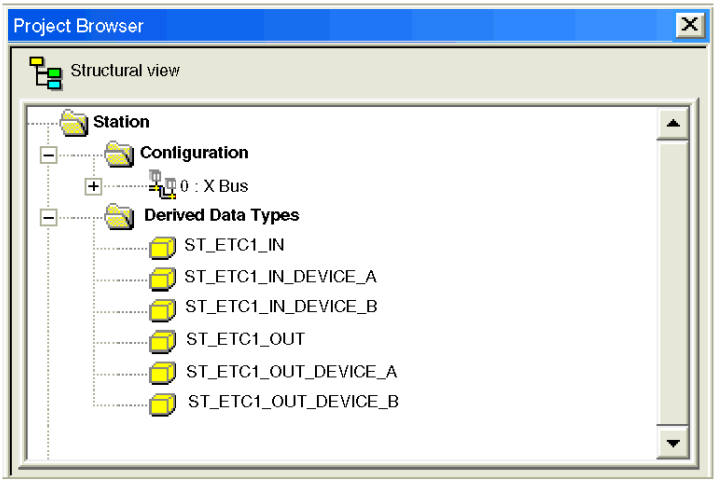
Use the EtherNet/IP configuration tool to configure:

- EtherNet/IP module channel properties *(see page 37)*
- EtherNet/IP module TCP/IP settings *(see page 50)*
- (optional) the EtherNet/IP module's local slave function *(see page 58)*
- properties for both:
  - Schneider-Electric remote devices *(see page 85)*
  - third-party remote devices *(see page 113)*

Creating or Updating Derived Data Types

After all EtherNet/IP module edits have been saved in the EtherNet/IP configuration tool, add these edits to the Unity Pro project, as follows:

Step	Action
1	Return to Unity Pro and select the **Configuration** page of the EtherNet/IP configurable server module, below. Note that the **Update application** button is now enabled. 
2	Click the **Update application** button.

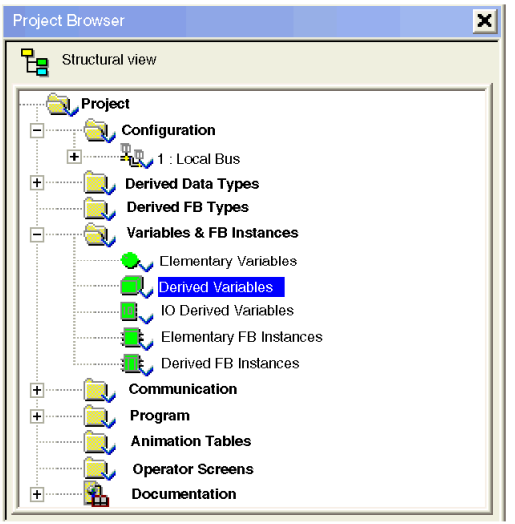
Step	Action
3	The Unity Pro software converts the EtherNet/IP configuration data to variables and derived data types, then prepares to import them into the Unity Pro project. If any conflict exists between a newly created item and an existing one, Unity Pro displays those conflicts in the **Import Trouble Report**, below:  In this example, the Unity Pro project configuration already includes the listed variables and derived data types.
4	If the **Import Trouble Report** opens, use its command buttons as follows: • **Keep:** Keeps the component of the current project. • **Replace:** Replaces the project component with the one from the import. • **Rename:** Renames the imported component, allowing you to keep both components.
5	After you have determined how to handle each imported item, click **OK**.
6	After you click **OK**, the **Project Browser** displays the new or edited derived data types, below: 

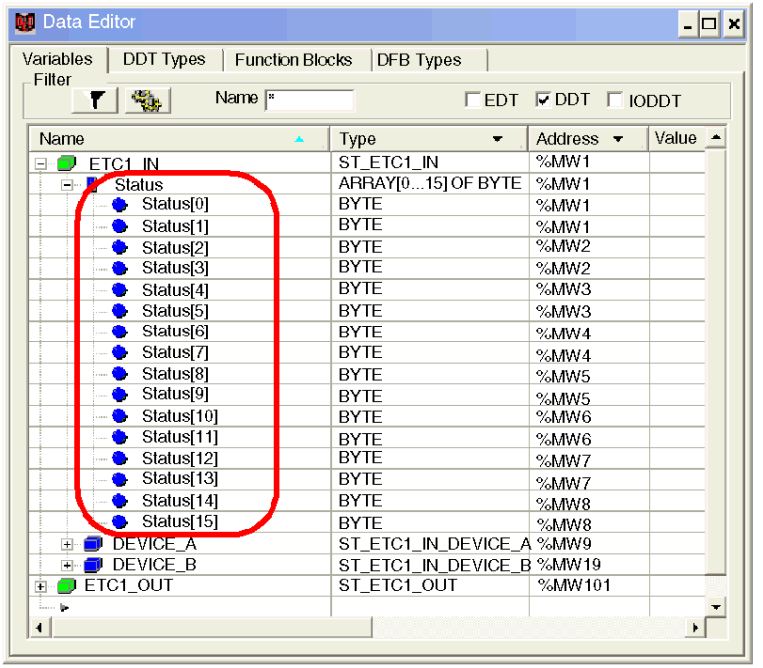
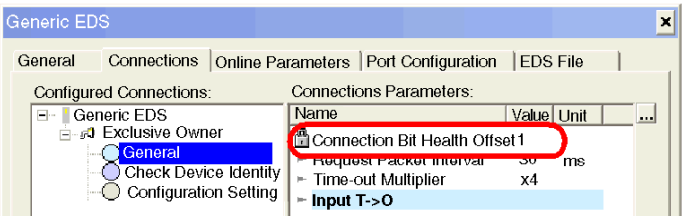
Viewing Derived Data Type Variables

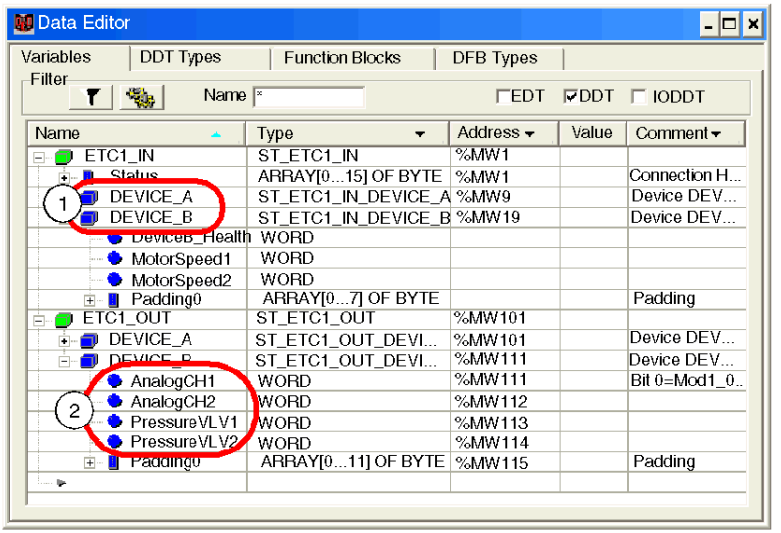
When you clicked on the **Update application** button, Unity Pro created a collection of derived data type variables. Use these variables to view the:

- status of all connections from the communication module to each remote device, where:
  - the status of all connections is displayed in an array of 16 bytes
  - each connection is represented by a single bit
  - a bit value of 1 indicates the connection is healthy
  - a bit value of 0 indicates the connection is lost, or the communication module can no longer communicate with the remote device
- value of input and output items you created using the Unity Pro EtherNet/IP configuration tool
- value of attributes defined by the EDS file of a remote device
- amount of padding, representing the reserved input or output memory space for a remote device

To view these derived data type variables:

Step	Description
1	Return to the main screen in Unity Pro.
2	In the **Project Browser** open the branch **Variables & FB Instances** and double-click on the **Derived Variables** sub-branch.  The **Data Editor** window opens, displaying the **Variables** tab. A check mark appears in the **DDT** checkbox. (If not, select the **DDT** checkbox.)

<table>
<tr><th>Step</th><th colspan="2">Description</th></tr>
<tr><td>3</td><td colspan="2">The Status variable—describing the status of all connections—is found beneath the first device in the first position. To display the Status variable's 16-byte array, expand the first device as depicted below
 
</td></tr>
<tr><td rowspan="3">4</td><td colspan="2">To determine which Status bit is mapped to a specific remote device:</td></tr>
<tr><td>a</td><td>In the Unity Pro EtherNet/IP configuration tool, open the Properties window for a remote device.</td></tr>
<tr><td>b</td><td>Open the Connections page, and click on the General node, below:
 
</td></tr>
<tr><td>c</td><td>In the above example, the Connection Bit Health Offset value of 1 maps to the first bit in the first byte of the Status variable, which can be represented as <code>Status[0].1</code>.</td></tr>
</table>

Step	Description
5	You can also use the **Data Editor** to display DDT variables. DDT variables are either: ● input and output items you created using the Unity Pro EtherNet/IP configuration tool, or ● attributes defined by the remote device's EDS file, or ● padding, representing the reserved but unused input or output memory space for a remote device The **Data Editor** presents DDT variables in separate input and output groups, sorted by device, as shown below:  **1** device names: user-created in the Unity Pro EtherNet/IP configuration tool **2** variable names: user-created as I/O items in the Unity Pro EtherNet/IP configuration tool, or defined as a property by the EDS file of the remote device

2.2 Using the Unity Pro EtherNet/IP Configuration Tool

Overview

This section describes the Unity Pro EtherNet/IP configuration tool user interface. Use the configuration tool to enter settings for the EtherNet/IP communication module and for other devices connected to your EtherNet/IP network.

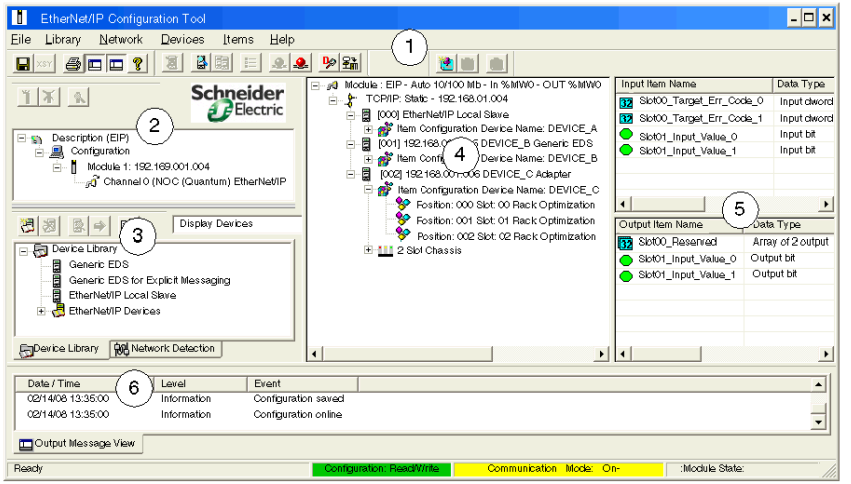
What Is in This Section?

This section contains the following topics:

EtherNet/IP Configuration Tool User Interface

Overview

The Unity Pro EtherNet/IP configuration tool user interface presents the following parts:



1 Main menu
2 Description area
3 Workspace area
4 Devices window
5 I/O area
6 Output Message window

The parts of the EtherNet/IP user interface are briefly described below.

Main Menu

The Main menu area consists of:

- A main menubar with the following menu items and commands:

Menu item	Contains commands for...
File	• file management and printing • GUI display selections • online / offline operations
Library	managing EDS files in the **Device Library**

Network	• automatic detection of EtherNet/IP network devices
	• online actions, including:
	  • explicit messaging
	  • pinging network devices
	• commissioning devices via port configuration settings
	• working with automatically detected devices in the **Network Detection** area

Devices	working with devices in the **Devices** window, including: • displaying devices in the Devices window tree control • commissioning devices via port configuration settings • creating and configuring CIP connections for devices • diagnosing devices
Items	adding, deleting, and renaming inputs and outputs in the I/O area
Help	• displaying versioning information of the Unity Pro EtherNet/IP configuration tool • online help

- 3 toolbars:

Toolbar	Contains commands that relate to...
Main toolbar	• file management and printing • GUI display selections
Devices toolbar	working with devices in the Devices window, including: • displaying devices in the Devices window tree control • commissioning devices • creating and configuring CIP connections for devices • diagnosing devices • online / offline operations
Items toolbar	adding, deleting, and renaming inputs and outputs in the I/O area

Description Area

The **Description** area describes the EtherNet/IP communication module and its IP address.

Workspace Area

The **Workspace** area consists of two tabs, containing the:

- Device Library, where you can:
  - view properties and EDS files for all available EtherNet/IP devices
  - add a new device and its EDS file to the Device Library
  - delete a device from the **Device Library**
  - manage the display of devices in the **Device Library** list
  - insert a selected device into the configuration in the **Devices** window
- Network Detection area, where you can:
  - automatically detect EtherNet/IP devices on the network
  - take online actions, including sending explicit messages and pinging network devices
  - view properties and EDS files for all available EtherNet/IP devices
  - insert a single selected device into the configuration in the **Devices** window
  - insert all detected device into the configuration in the **Devices** window, replacing all devices in the configuration

You can show or hide the workspace area using the **File** →**Preferences** →**Workspace** command.

Devices Window

The Devices window contains a tree control, containing all devices that have been added to your EtherNet/IP network configuration. In the **Devices** window, you can:

- display and edit the properties of selected EtherNet/IP devices, including:
  - EtherNet/IP communication modules
  - local slaves
  - remote devices
  - I/O modules
- commission devices
- create and configure CIP connections for devices
- open the I/O area and display individual inputs and outputs
- diagnose device connections

I/O Area

The I/O Area displays the configuration data for each input and output, including the:

- name
- data type
- offset within the device
- offset within the connection
- address where the I/O data is sent to, or sent from

The I/O area is displayed only when a device I/O connection is selected in the Devices window configuration.

Output Message Window

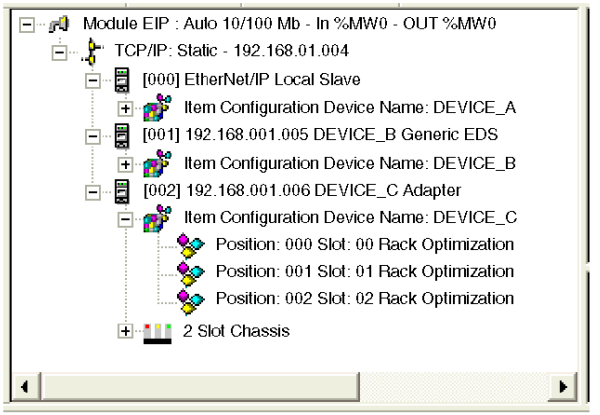
The **Output Message** window displays a sortable list of events. The Output Message window can be configured to show or hide each item's:

- date and time
- event level:
  - informational
  - warning
  - error

Devices Window

Overview

The **Devices** window is located in the center of the EtherNet/IP configuration tool's user interface and displays a node for each device in your network configuration. A example of the **Devices** window appears below:



Configurable Properties

The **Devices** window displays a node for each device—and for each device's configurable property group— in your network configuration. Each node is identified by an icon, as follows:

Node	Icon	This node is used to configure...
Channel		The properties of the EtherNet/IP module's communication channel.
TCP/IP		The EtherNet/IP communication module's IP addressing, SNMP and DHCP server settings.
Local slave		Properties related to the module's role as an I/O adapter to a remote device acting in the role of I/O scanner.
Device		The properties of any EtherNet/IP network device with an IP address, including both modular and non-modular devices.
Items collection		The name assigned to a group of I/O items.

Node	Icon	This node is used to configure...
Item		The properties of a CIP connection between the EtherNet/IP communication module and individual I/O items. If the type of connection is: ● **rack optimized**: click on the connection in the first position to display all rack optimized I/O items. ● **direct**: click on a connection for any position to display the I/O items for that connection.
Chassis		The properties of a chassis that is part of a modular device.
Module		The parameters of an I/O module that is part of a modular device.

Configuring Properties in the Devices Window

Overview

Use the **Devices** window, in the Unity Pro EtherNet/IP configuration tool, to display and configure properties for the EtherNet/IP communication module and other devices on your EtherNet/IP network.

To configure properties, double-click on the Devices window node associated with the properties *(see page 33)* you want to configure.

For example, to configure the EtherNet/IP communication module network channel properties, double-click on the channel icon to display the **Channel Properties** window. When the window first opens, it displays 2 tabbed pages:

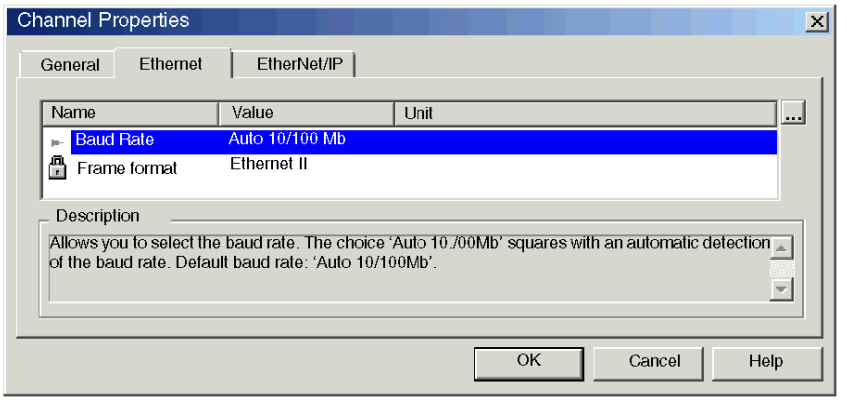
- General (the default tab)
- Ethernet

Note that the Channel Properties window can also display 2 additional pages:

- Ethernet/IP, by operating in Advanced mode (**File →Preferences →Advanced**)
- Module Information, by operating online (**File →Go Online**)

Displaying Property Values

Most property windows let you display a description of a selected property. Select a property in the **Name** column to display a brief description of the selected property in the **Description** area at the bottom of the window:



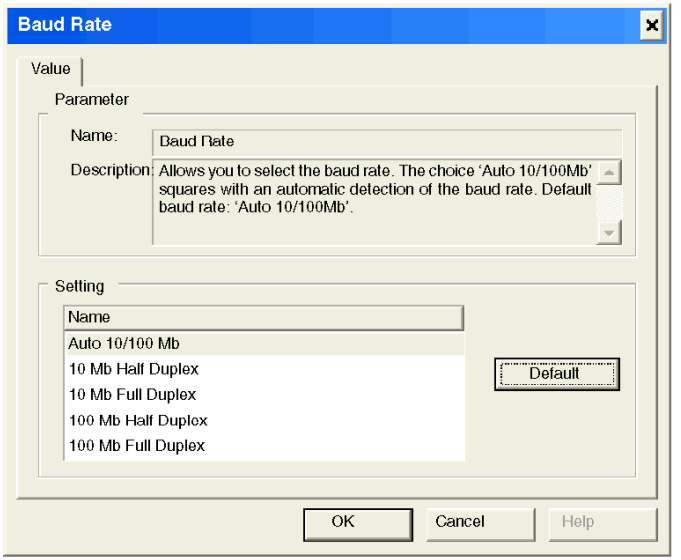
Property Types

Properties can be either read/write or read-only, as indicated by its icon:

This icon...	Indicates the property is...
	Read-only. This property value is locked and cannot be edited.
	Read-write. This property value can be edited.

Editing Property Values

To edit a read-write property value, follow these steps:

Step	Action
1	Select a read-write property.
2	There are two ways to perform an edit: • Double-click the left mouse button on the property name. • Click the ellipsis (...) button located at the right of the list header bar, then select **Properties** in the popup menu. A window opens—in this case for the **Baud Rate** property—where you can edit the parameter value:  **Note:** Some other properties are editable by typing in a value within a stated range.
3	After completing your edits click **OK** to close the **Properties** window and save your edits.

2.3 Configuring Network Channel Properties

Overview

This section describes how to configure network channel properties with the EtherNet/IP configuration tool.

What Is in This Section?

This section contains the following topics:

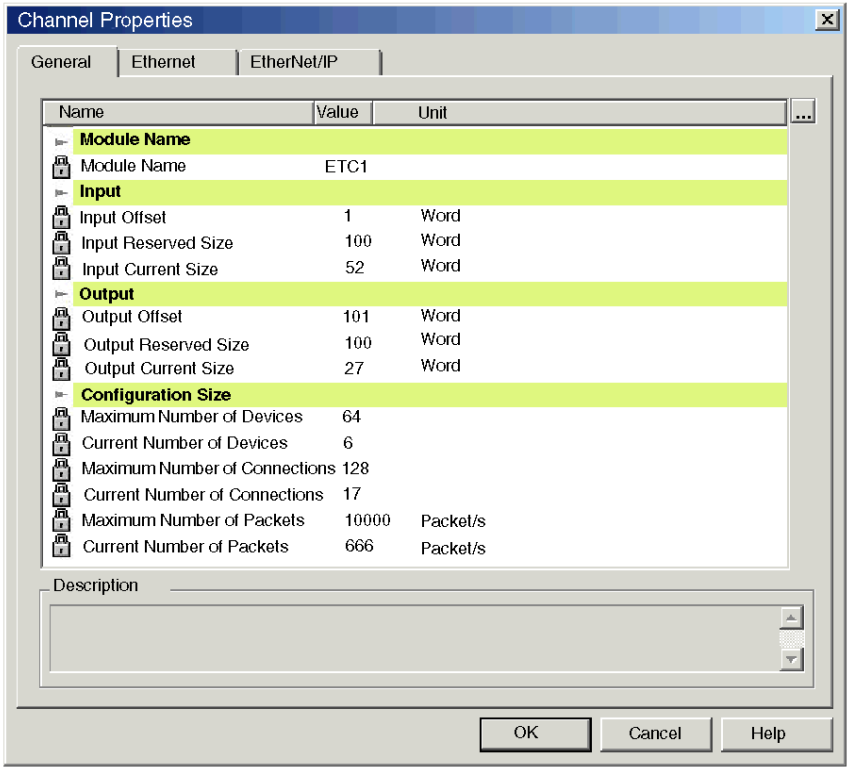
Configuring Channel Properties: The General page

The General Page

The **General** page of the **Channel Properties** window displays read-only properties that describe the:

- EtherNet/IP module name
- size and location of inputs and outputs
- size of the EtherNet/IP configuration

The properties are determined by the communication module's EDS file, the configuration design, and settings entered in the **Configuration** page of Unity Pro for the communication module.



Properties

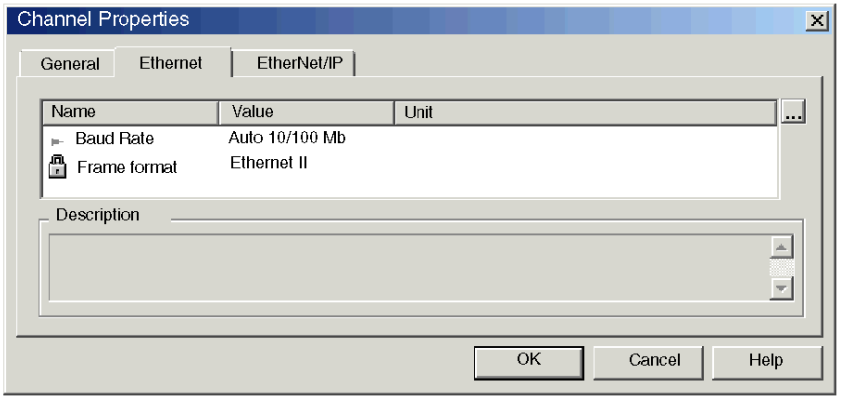
Name	Description	Value set by...
Module Name		
Module Name	The name of the EtherNet/IP module	**Configuration** page in Unity Pro
Input		
Input Offset	The starting address for inputs (%MW index)	**Configuration** page in Unity Pro
Input Reserved Size	The total number of words configured for inputs (Max size)	**Configuration** page in Unity Pro
Input Current Size	The actual number of inputs used in the application	network design in the configuration tool's **Devices window**
Output		
Output Offset	The starting address for outputs (%MW index)	**Configuration** page in Unity Pro
Output Reserved Size	The total number of words configured for outputs (Max size)	**Configuration** page in Unity Pro
Output Current Size	The actual number of outputs used in the application	network design in the configuration tool's **Devices window**
Note: When configuring an offset and a reserved size for both inputs and outputs, be sure that inputs and outputs do not overlap.		
Configuration Size		
Maximum Number of Devices	The maximum number of devices that can be added to the configuration.	predefined
Current Number of Devices	The number of devices currently in the configuration.	network design in the configuration tool's **Devices window**
Maximum Number of Connections	The maximum number of connections that can be managed by the module.	predefined
Current Number of Connections	The number of connections in the configuration.	network design in the configuration tool's **Devices window**
Maximum Number of Packets	The maximum number of packets the module is able to manage.	predefined
Current Number of Packets	The number of packet/s that will be generated by the current configuration.	network design in the configuration tool's **Devices window**

Configuring Channel Properties: The Ethernet page

The Ethernet Page

Use the **Ethernet** page of the **Channel Properties** window to:

- view and edit the **Baud Rate**
- view the **Frame format**



NOTE: Refer to the topic Configuring Properties in the Devices Window *(see page 35)* for information on how to display property descriptions and edit property values.

Properties

Name	Description	Type
Baud Rate	The transmission speed and duplex mode for the configuration. To change these settings, double-click on the field name and select one of the following: ● Auto 10/100 Mb (the default) ● 10 Mb Half duplex ● 10 Mb Full duplex ● 100 Mb Half duplex ● 100 Mb Full duplex **Note:** The default setting—**Auto 10/100 Mb**—is recommended. It causes the connected devices to perform auto-negotiation and thereby determine the fastest common transmission rate and duplex mode.	Read-Write
Frame Format	Ethernet II is the only frame format available for this module.	Read-Only

Configuring Channel Properties: The EtherNet/IP page

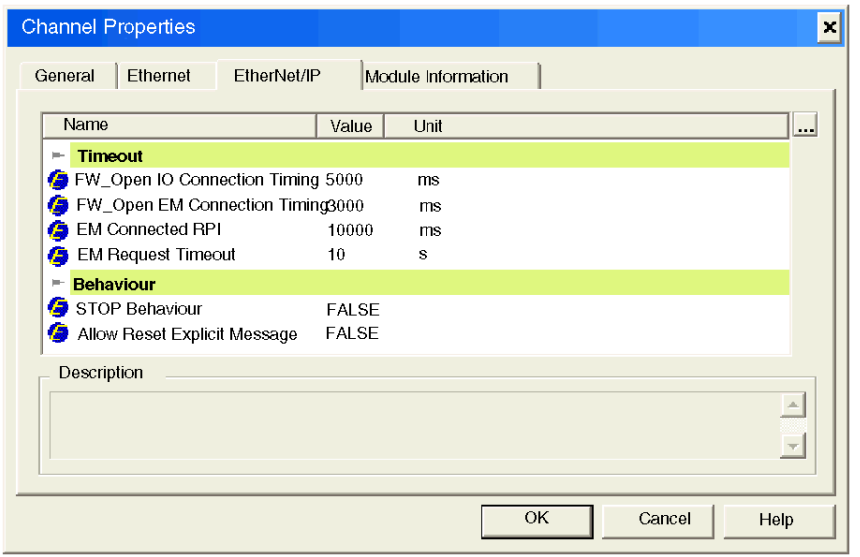
The EtherNet/IP Page

Use the **EtherNet/IP** page of the **Channel Properties** window to configure:

- properties that determine how the EtherNet/IP communication module, in its role as an I/O scanner, opens both implicit and explicit connections
- the frequency for transmitting produced data over implicit connections
- the timeout period for explicit connections
- the behavior of the module, in its role as an I/O scanner, when:
  - the application is stopped, or
  - the EtherNet/IP module receives a reset service request

NOTE: This page is displayed only when you are using Advanced Mode. Advanced mode properties are indicated by the icon.

To turn on Advanced Mode, select: **File** →**Preferences** →**Advanced**



NOTE: Refer to the topic Configuring Properties in the Devices Window *(see page 35)* for information on how to display property descriptions and edit property values.

Configuring EtherNet/IP Properties

Note: Only an experienced developer of EtherNet/IP networks should edit any of the following read-write properties.

Name	Description
Timeout	
FW_Open IO Connection Timing	The amount of time the EtherNet/IP module waits for the Forward_Open IO messaging transaction to open an implicit messaging connection. Default = 5000 ms
FW_Open EM Connection Timing	The amount of time the EtherNet/IP module waits for the Forward_Open IO messaging transaction to open an explicit messaging connection. Default = 3000 ms
EM Connected RPI	The value used to set the T->O (target to originator) and O->T (originator to target) requested packet interval (RPI) for all explicit message connections. This value is used to calculate the lifetime of a connection. Default = 10000 ms.
EM Request Timeout	The amount of time the EtherNet/IP module waits between a request and reply of an explicit message. Default =10 s.
Output	
STOP Behavior	The state of the EtherNet/IP module when the CPU application goes into a STOP state: ● TRUE indicates that the module enters STOP state (implicit connections are closed). ● FALSE indicates that the module enters IDLE state (implicit connections are not closed). Default = FALSE
Allow Reset Explicit Message	The behavior of the EtherNet/IP module—as I/O scanner—when it receives a reset service request: ● TRUE indicates the module resets itself. ● FALSE indicates the module ignores the reset service request and continues uninterrupted operations. Default = FALSE

Configuring Channel Properties: The Module Information page

The Module Information Page

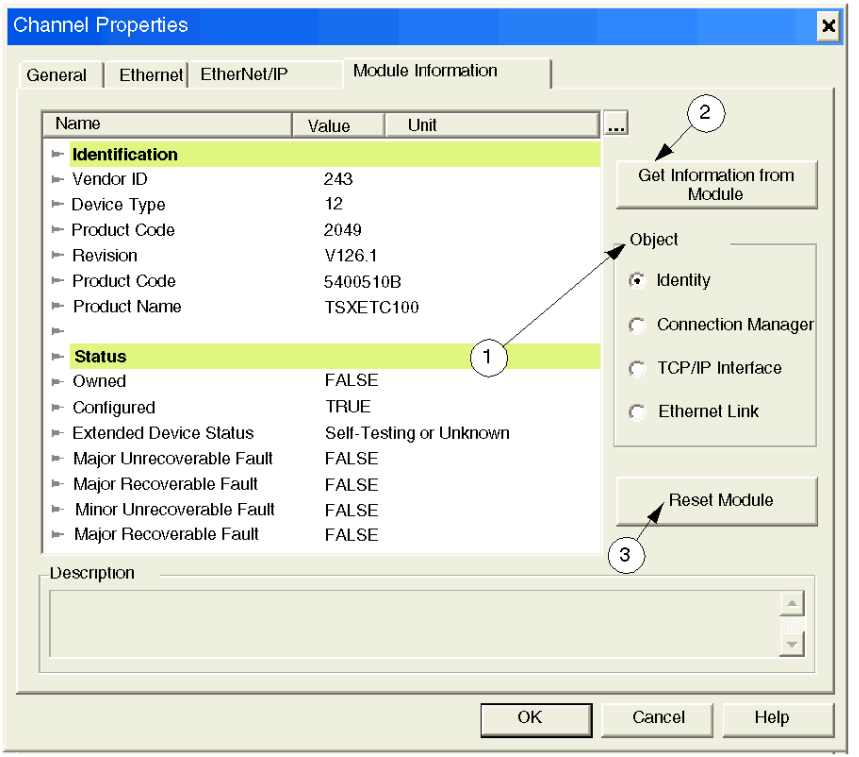
Use the **Module Information** page of the **Channel Properties** window to display properties obtained from the EtherNet/IP communication module. In this page you can:

- retrieve data from the communication module
- display retrieved module data by a selected object group, including data relating to the module's:
  - Identity
  - Connection Manager
  - TCP/IP Interface
  - Ethernet Link
- refresh data

NOTE:

- This page is displayed only when the Unity Pro EtherNet/IP configuration tool is operating online. To operate online, select **File** →**Go Online**.
- All object groups are displayed only when you are operating in Advance mode. To operate in Advance mode, select **File** →**Preferences** →**Advance**.

Displaying module information is a 3-step process, as described below:



NOTE: Refer to the topic Configuring Properties in the Devices Window *(see page 35)* for information on how to display property descriptions and edit property values.

Step 1	Select a property type in the **Object** list: ● identity ● Connection Manager ● TCP/IP Interface ● Ethernet Link
Step 2	Click the **Get Information from Module** button to populate property data.
Step 3	Periodically click the **Reset Module** button to update property data.

Identity Properties and Status

After selecting **Identity**, the following information is displayed.

Property	Description
Identification	
Vendor ID	243
Device Type	12
Product Code	2
Revision	The revision number of the device
Serial Number	The serial number of the device.
Product Name	TSX ETC 100
Status	
Owned	A TRUE setting indicates that the device (or an object within the device) has an owner. The setting of this bit means that the Predefined Master/Slave Connection Set has been allocated to a master.
Configured	A TRUE setting indicates that the application of the device has been configured to do something different than the out-of-the-box default. This does not include configuration of the communications.
Extended Device Status	The vendor-specific or already defined status.
Major Unrecoverable Fault	A TRUE setting indicates the device detected a problem with itself, which caused the device to go into the Major Unrecoverable Fault state.
Major Recoverable Fault	A TRUE setting indicates the device detected a problem with itself, which caused the device to go into the Major Recoverable Fault state.
Minor Unrecoverable Fault	A TRUE setting indicates the device detected a problem with itself, which is thought to be unrecoverable. The problem does not cause the device to go into one of the faulted states.
Minor Recoverable Fault	A TRUE setting indicates the device detected a problem with itself, which is thought to be recoverable. The problem does not cause the device to go into one of the faulted states.

Connection Manager Properties

After selecting **Connection Manager**, the following information is displayed.

Property	Description
Open Counters	
Open Requests	The number of forward open service requests received.
Format Rejects	The number of forward open service requests which were rejected due to bad format.
Resource Rejects	The number of forward open service requests which were rejected due to lack of resources.
Other Rejects	The number of forward open service requests which were rejected for reasons other than bad format or lack of resources.
Close Counters	
Close Requests	The number of forward close service requests received.
Format Rejects	The number of forward close service requests which were rejected due to bad format.
Other Rejects	The number of forward close service requests which were rejected for reasons other than bad format.
Other Counters	
Connection TimeOuts	The total number of connection timeouts that have occurred in connections controlled by this Connection Manager
Numbers of Connection	The number of connections.

TCP/IP Interface Properties

After selecting **TCP/IP Interface**, the following information is displayed. Not all properties apply to the communication module.

Property	Description
Status	Indicates the status of the configuration: ● 0 = not configured ● 1 = a valid configuration acquired from BOOTP or nonvolatile storage
Configuration Capability	● BOOTP Client Indicates that the device is capable of acquiring its network configuration via BOOTP. ● Configuration Settable Indicates that the configuration is settable.
Startup Configuration	Determines how the device acquires its initial configuration at startup. **Note:** If the device was previously configured, it uses the previously stored interface configuration values.

Property	Description
IP Address	The device IP address. A 0.0.0.0 address indicates an IP address has not been configured.
Network Mask	The device network mask. A 0.0.0.0 address indicates a network mask address has not been configured.
Gateway Address	The default gateway address. A 0.0.0.0 address indicates a gateway address has not been configured.
Primary Name Server Address	(not applicable)
Secondary Name Server Address	(not applicable)
Domain Name	(not applicable)
Host Name	(not applicable)
Safety Network Number	(not applicable)
TTL Value	The value that the device uses for the IP header's Time-to-Live field when sending packets via IP an multicast.
Multicast Address Allocation Control	This determines how the device shall allocate IP multicast addresses. If set to: **0 -** Multicast addresses are generated using the default allocation algorithm. **1 -** Multicast addresses are allocated according to the values specified in the two following parameters.
Number of IP Multicast Addresses Allocated	The number of IP multicast addresses that are allocated.
Starting Multicast IP Address	The starting multicast address from which allocation begins.

Ethernet Link Properties

After selecting **Ethernet Link**, the following information is displayed.

Property	Description
General	
Interface Speed	The interface speed currently in use. A 0 is shown if the speed has not been determined.
Link Status	Indicates whether or not the Ethernet communications interface is connected to an active network.
Duplex Mode	Indicates that duplex mode currently in use.

Property	Description
Negotiation Status	Indicates the status of link auto-negotiation. If set to: **0 -** Auto-negotiation in progress. **1 -** Auto-negotiation and speed detection has failed. Default values for speed and duplex are being used. **2 -** Auto negotiation has failed but the speed has been detected. Duplex was defaulted. The default value is product-dependent; recommended default is half duplex. **3 -** Successfully negotiated speed and duplex. **4 -** Auto-negotiation was not attempted. Speed and duplex has been forced.
Manual Setting Requires Reset	If set to: **0 -** The interface can activate changes to link parameters (auto-negotiate, duplex mode, interface speed) automatically. **1 -** The device requires a reset service be issued to its Identity Object in order for the changes to take effect.
Local hardware Fault	A local hardware fault.
Physical Address	The MAC layer address.
Input	
Octets	The number of octets received on the interface.
Ucast Packets	The number of subnetwork-unicast packets delivered to a higher-layer protocol.
NUcast Packets	The number of non-unicast packets delivered to a higher-layer protocol.
Discards	The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol.
Errors	The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol.
In Unknown Protocols	The number of packets received via the interface which were discarded because of an unknown or unsupported protocol.
Output	
Octets	The number of octets sent on the interface.
Ucast Packets	The total number of packets that higher-level protocols requested be transmitted to a subnetwork-unicast address.
NUcast Packets	The total number of packets that higher-level protocols requested be transmitted to a non-unicast address.
Discards	The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted.
Errors	The number of outbound packets that could not be transmitted because of errors.

Property	Description
Error Counters	
Alignment Errors	The number of frames received on this interface that are not an integral number of octets in length and do not pass the FCS check.
FCS Errors	The number of frames received on this interface that are an integral number of octets in length but do not pass the FCS check.
Single Collisions	The number of successfully-transmitted frames on this interface for which transmission is inhibited by exactly one collision.
Multiple Collisions	The number of successfully-transmitted frames on this interface for which transmission is inhibited by more than one collision.
SQE Test Errors	The number of times a SQE test error message has been generated.
Deferred Transmissions	The number of frames for which the first transmission attempt on this interface has been delayed because the medium is busy.
Late Collisions	The number of times a collision is detected later than 512 bit-times into the transmission of a packet.
Excessive Collisions	The number of frames for which transmission on this interface has failed due to excessive collisions.
MAC Transmit Errors	The number of frames for which transmission on this interface has failed due to an internal MAC sublayer transmit error.
Carrier Sense Errors	The number of times that the carrier sense condition was lost or never asserted when attempting to transmit a frame on this interface.
Frame Too Long	The number of frames received on this interface that exceeded the maximum permitted frame size.
MAC Receive Errors	The number of frames for which reception on the interface has failed due to an internal MAC sublayer receive error.

2.4 Configuring the TCP/IP Address Settings

Overview

This section provides information about how to configure the TCP/IP address settings for the EtherNet/IP communication module.

What Is in This Section?

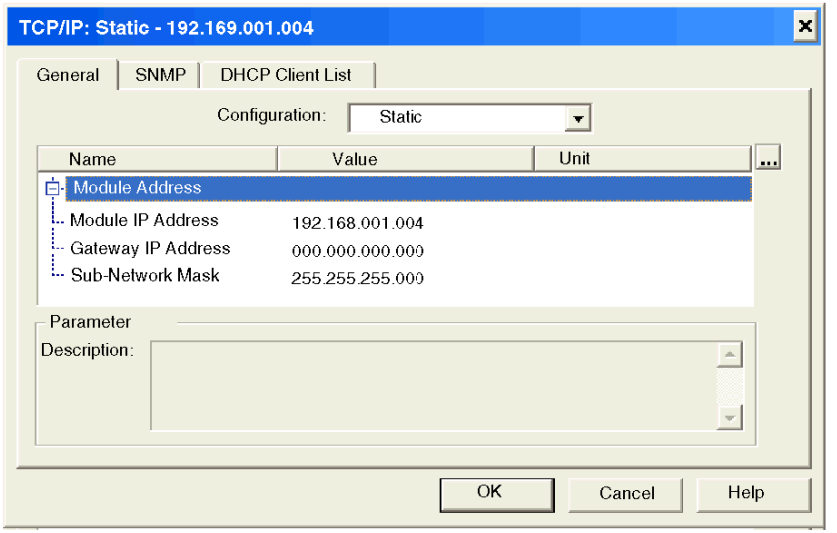
This section contains the following topics:

TCP/IP Properties: The General Page

The General Page

Use the **General** page of the **TCP/IP** properties window to configure the IP address of the EtherNet/IP communication module.

Open the **TCP/IP** properties window by clicking on the TCP/IP icon in the **Devices** window.



NOTE: Refer to the topic Configuring Properties in the Devices Window *(see page 35)* for information on how to display property descriptions and edit property values.

Selecting a Configuration Mode

Use the **Configuration** list to specify a configuration mode. The configuration mode setting determines how the module obtains its IP address at startup. Choices are:

Configuration Mode	Description
Static	The module uses the module IP address, gateway IP address, and sub-network mask configured in this page.
Flash Memory	The module uses the IP address configured via the TCP/IP object and stored flash memory. An IP address configured by this process survives a warm re-start (during which power to the device is continuously maintained), but is lost in the case of a cold re-start (where power to the device is turned off for a time).
BOOTP	The module uses an IP address assigned by a BOOTP server.

Setting the Module Addresses in Static Mode

Three IP address properties need to be configured for the EtherNet/IP communication module in static configuration mode:

Property	Description
Module IP Address	The 32-bit identifier—consisting of both a network address and a host address—assigned to a device connected to a TCP/IP Internet network using the Internet Protocol (IP).
Gateway Address	The address of a device, if any, that serves as a gateway to the EtherNet/IP module.
Sub-Net Mask	The 32-bit value used to hide (or mask) the network portion of the IP address and thereby reveal the host address of a device on a network using the IP protocol.

Default Address Configurations

The module uses a default address configuration when it is not configured or when a duplicate IP address is detected. The default address is based on the MAC address of the module and makes it possible for several Schneider devices to use their default network configuration on the same network.

The module uses the following default address configurations.

- Default IP Address
  This default address starts with 10.10 and uses the last two bytes of the MAC address. As an example, a device with the MAC address of 00:00:54:10:8A:05 has a default IP address of 10.10.138.5 (0x8A=138, 0x05=5).
- Default Subnet Mask
  The default address is 255.0.0.0 (a class A mask).
- Default Gateway Address
  The default gateway address is identical to the default IP address.

Duplicate IP Address Checking

Before going online, the module sends out at least four ARP (Address Resolution Protocol) messages with a proposed IP address.

- If an answer is returned
  - There is a device already using the IP address.
  - The module will not use the proposed IP address and uses the default IP address.
- If an answer is not returned
  - The module uses the IP address (along with the associated network parameters.)

TCP/IP Properties: Configuring the SNMP Agent

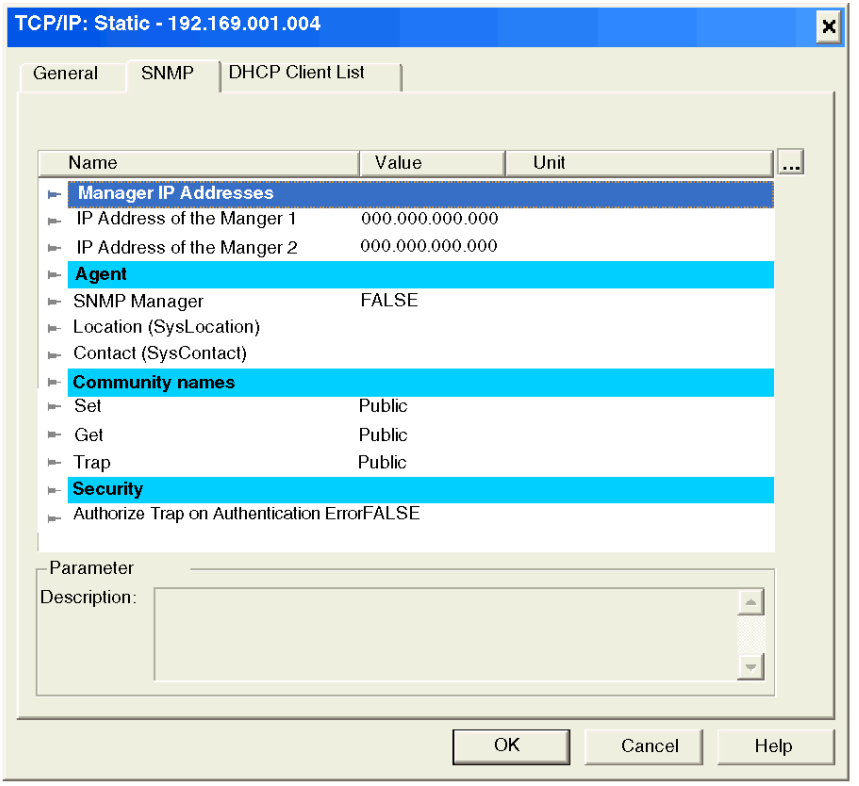
The SNMP page

Use the SNMP page of the TCP/IP properties window to configure the SNMP agent *(see page 135)* in the EtherNet/IP communication module. An SNMP agent is a software component that reports management data about the module to another device acting as an SNMP manager.

The SNMP agent can connect to and communicate with up to 2 SNMP managers as part of an SNMP service. The SNMP service includes:

- authentication checking, by the EtherNet/IP communication module, of any SNMP manager that sends SNMP requests
- management of event, or trap, reporting by the module

Click on the SNMP tab to access the SNMP agent window:



NOTE: Refer to the topic Configuring Properties in the Devices Window *(see page 35)* for information on how to display property descriptions and edit property values.

Viewing and Configuring SNMP Properties

The following properties can be viewed and edited in the SNMP page:

Property	Description
Manager IP Addresses:	
IP Address of the Manager 1	The IP address of the first SNMP manager to which the EtherNet/IP module's SNMP agent sends notices of traps.
IP Address of the Manager 2	The IP address of the second SNMP manager to which the module's SNMP agent sends notices of traps.
Agent:	
SNMP Manager	Select either: • **TRUE**: the Location and Contact information is provided by a network management tool • **FALSE**: Location and Contact settings are made in this window
Location	The device location (32 characters maximum)
Contact	Information describing the person to contact for device maintenance (32 characters maximum)
Community Names:	
Get	Password required by a MIB-II SNMP agent authorizing read commands from an SNMP manager. Default = **Public**.
Set	Password required by a MIB-II SNMP agent authorized write commands from an SNMP manager. Default = **Public**
Trap	Password a MIB-II SNMP manager requires from an SNMP agent causing the SNMP manager to accept trap notices from the SNMP agent. Default = **Public**
Security:	
Authorize Trap on Authentication Error	Causes the SNMP agent to send a trap notice to the SNMP manager if an unauthorized manager sends a Get or Set command to the agent. Default = **FALSE**.

TCP/IP Properties: Configuring the DHCP Server

The DHCP Client List Page

The EtherNet/IP communication module can be configured to perform the function of DHCP server. Connected network devices can subscribe to this DHCP service and obtain their IP parameters from the module.

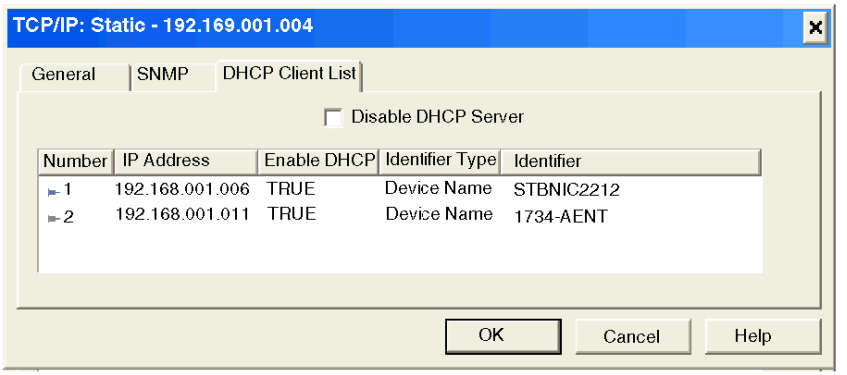
Use this page to:

- enable and disable the DHCP service, and
- view a list of all network devices indicating whether each connected network device does—or does not—subscribe to the DHCP service

NOTE: The DHCP service is not enabled or disabled for a specific network device in this page. See the topic Enabling the DHCP Service *(see page 56)*, below, for information on how to enable the DHCP service for a specific device.

Viewing the DHCP Client List

The **DHCP Client List** includes a row for each networked EtherNet/IP device, identifying the devices that have subscribed to the DHCP service:

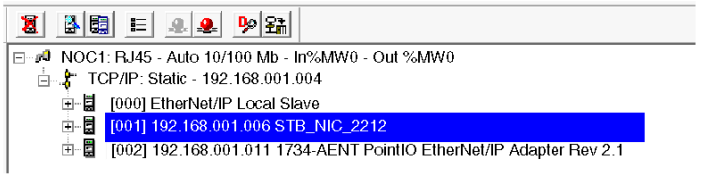
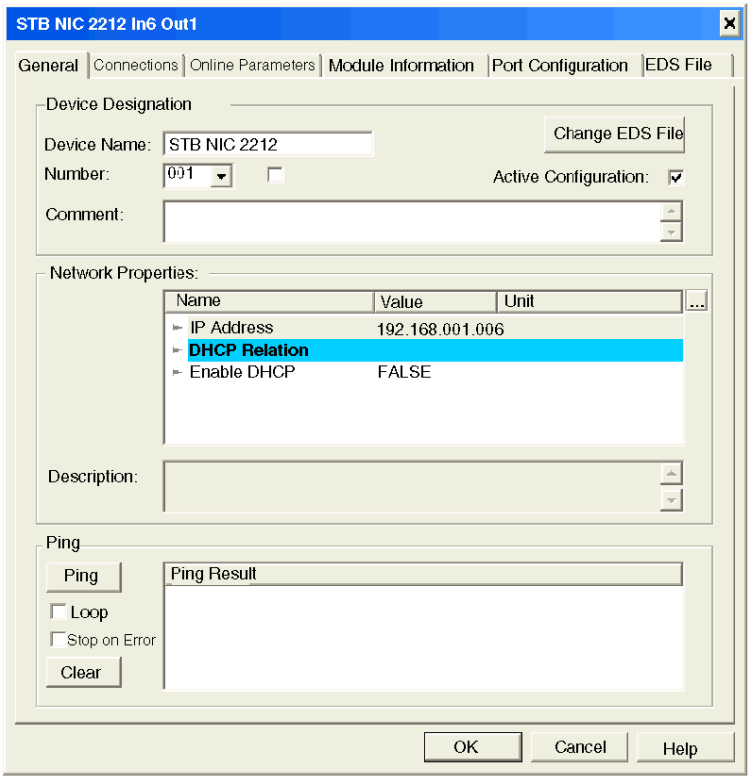


The list contains the following information for each networked device:

Property	Description
Number	The number assigned to the device in the EtherNet/IP configuration tool.
IP Address	The device IP address associated with the device.
Enable DHCP	TRUE indicates that the device subscribes to the DHCP service.
Identifier Type	Indicates the mechanism used by the server to recognize the client (MAC address or DHCP device name).
Identifier	The actual MAC address or DHCP device name.

Enabling the DHCP Service

The DHCP service for an EtherNet/IP device is not enabled in this page. Instead it is enabled and disabled in the remote EtherNet/IP device configuration. To turn on the DHCP service for a specific device, follow these steps:

Step	Action
1	n the Unity Pro EtherNet/IP configuration tool, select the DHCP client device in the **Devices** window. In this example, the selected client is an STB_NIC_2212: 
2	Select **Devices** →**Properties**. The **General** page of the **Properties** window opens for the selected device, indicating the DHCP client service is disabled (the default setting). 

Step	Action	
3	In the **Network Properties** area, under the heading **DHCP Relation**, configure the following properties:	
	Property:	**Action:**
	Enable DHCP	Select **TRUE**.
	DHCP Client Identifier	Select either: • **MAC Address**, or • **Device Name**
	Mac Address/Device Name	Enter a value for either the device name or the MAC Address.
4	Click **OK** to close the device's **Properties** window and save your edits.	

2.5 Configuring the EtherNet/IP Communication Module as an I/O Adapter

Overview

This section describes how to configure the EtherNet/IP communication module as an I/O adapter (local slave). In this role, the module initiates no messages. Instead, it responds to:

- implicit messaging requests from a remote device for periodic data, at the established RPI rate, and
- explicit messaging requests from other EtherNet/IP devices on the network

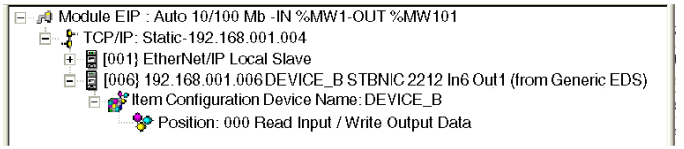
What Is in This Section?

This section contains the following topics:

Identifying the Local Slave

Overview

When the Unity Pro EtherNet/IP configuration tool first opens, it automatically includes a Local Slave node in the **Devices** window:



Key Features

Features	Description
Types of connection	• Multicast Point to point is supported in both directions: O->T (Originator to Target) and T->O (Target to Originator) • Real time format 32 bit run/idle header, zero data length, none and heartbeat • Trigger T->O (Target to Originator) cyclic
Sizes	• Input sizes From 1 to 505 bytes • Output sizes From 1 to 509 bytes • Configuration size 0 words (read-only)

Local Slave Inputs and Outputs

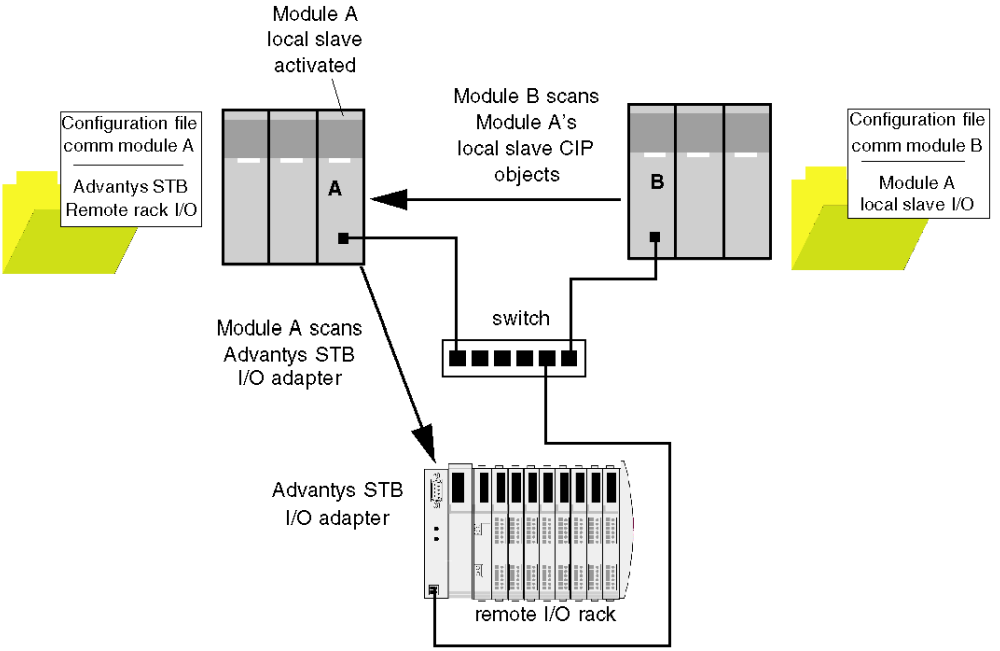
The EtherNet/IP communication module can be used as an I/O adapter. To enable this functionality, select **Active Configuration** in the **Local Slave** properties window *(see page 62)*.

When the local slave function of an EtherNet/IP communication module is enabled, the module's CIP objects *(see page 173)* are exposed to, and can be accessed by, other EtherNet/IP devices.

The I/O data exchange, between the remote device and the local slave, is configured as part of the remote scanning module's configuration settings.

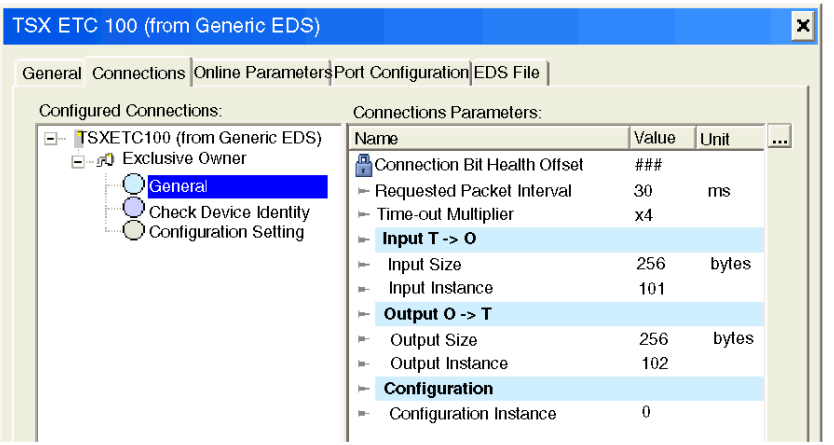
In the following example:

- module A acts as both:
  - an I/O scanner of the Advantys STB I/O adapter, and
  - an I/O adapter, with its CIP objects accessible to remote EtherNet/IP devices
- module B acts as an I/O scanner of the local slave function of module A. Module B can access the exposed CIP objects of module A. The I/O data exchange between module B and module A is configured in the settings for module B.



Configuring the Connection

The I/O data exchange between module B (in its role as an I/O scanner) and module A (in its role as an I/O adapter) is configured in the settings for module B. Do this in the **Connections** page of the remote EtherNet/IP communication module—here, module B—**Properties window**:



Configuring the I/O items

You can configure input and output items in groups of 1 or more single bits, 8-bit bytes, 16-bit words, 32-bit dwords, or 32-bit IEEE floating values. The number of items you create depends upon the data type and size of each item.

The process for creating and defining I/O items for the local slave are the same as for any I/O adapter, and depend upon the type of item you wish to create.

For an I/O configuration example, see the how the following I/O items were configured for the STB NIC 2212 network interface module:

- discrete input items *(see page 101)*
- numeric input items *(see page 107)*
- discrete output items *(see page 104)*
- numeric output items *(see page 110)*

Configuring Local Slave Properties: The General page

The General Page

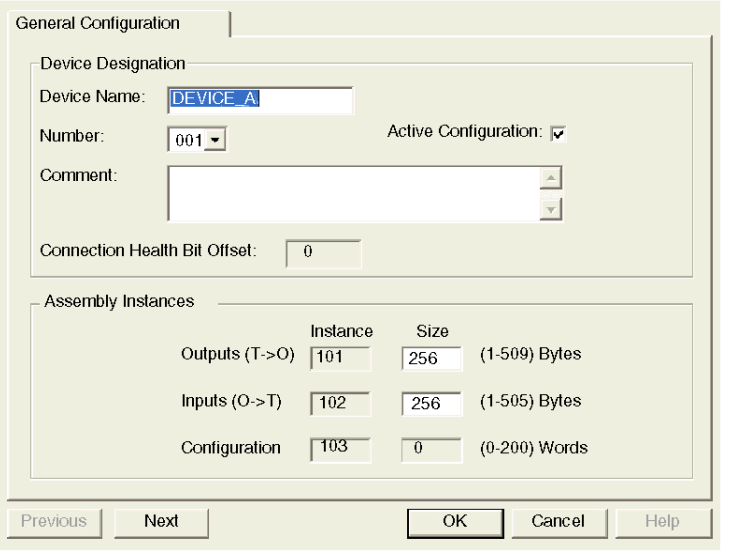
Use the **General** page to configure the EtherNet/IP communication module to serve as an I/O adapter to a remote device.

The following steps describe a sample configuration of the local slave function. Your configuration may be different.

Configuring the Local Slave

To configure the local slave function, follow these steps:

Step	Action
1	In the **Devices** window, right click on the **EtherNet/IP Local Slave** icon, then select **Properties** in the popup menu.  The **General** page of the **Local Slave** properties window opens.

Step	Action
2	Enter settings (refer to the table below) for the following parameters to configure local slave functionality for the EtherNet/IP module.  **Note:** When using explicit messaging to read the EtherNet/IP module's assembly object, be sure to allocate sufficient room for the response, because the size of the response will equal the sum of: the size of the assembly + Reply service (1 byte) + General Status (1 byte) Local slave properties are described, below.
3	The next task is to configure slave inputs and outputs.

Local Slave Properties

The following property settings have been made in this example:.

Setting	Description
Device Designation section:	
Active Configuration	• A **selected** checkbox indicates the local slave service is enabled. • A **de-selected** checkbox indicates the local slave service is disabled and the current local slave service settings are saved. In this example, this setting is **selected**.
Device Name	Assign the local slave a unique name, consisting of up to 32 characters, including numbers, letters, and the underscore character. In this example, the auto-generated name **DEVICE_A** is accepted.
Number	The unique number—or identifier—assigned to the device. In this example, select the number **001**.
Comment	User-defined free text comment area. 80 characters maximum. In this example, leave blank.
Connection Health Bit Offset	Auto-generated integer (0...127) indicating the offset of the connection's health bit in the status byte array of the input area. **Note:** This setting is auto-generated only when the local slave settings are input and the network configuration is saved.
Assembly Instances section: • O indicates the originator—or I/O scanner—device • T indicates the target—or I/O adapter—device	
Outputs T -> O Instance	A read-only value always set to 101.
Outputs T -> O Size	The maximum size reserved for local slave outputs, in bytes. An integer from 0...509. In this example, accept the default of **256**.
Inputs O -> T Instance	A read-only value always set to 102.
Inputs O -> T Size	The maximum size reserved for local slave inputs, in bytes. An integer from 0...509. In this example, accept the default of **256**.
Configuration Instance	A read-only value always set to **103**.
Configuration Size	A read-only value always set to **0**.

Adding Devices to an EtherNet/IP Network

3

Overview

This chapter presents examples of how to add devices to, and how to configure these device for operations on, your EtherNet/IP network.

What Is in This Chapter?

This chapter contains the following sections:

3.1 Adding Devices to an EtherNet/IP Network

Effect of Device Position on Input and Output %MW Memory Addresses

Introduction

The Unity Pro EtherNet/IP configuration tool assigns a %MW memory address to the Inputs and outputs of a remote device, or a local slave, when it is activated.

By default:

- a remote EtherNet/IP device is activated when it is added to an EtherNet/IP network, but
- the EtherNet/IP communication module's local slave function is not activated when it is automatically added to a newly created network—instead, it must be manually activated

This topic describes:

- the effect of activating the local slave on the %MW memory address assignment for inputs and outputs of previously configured EtherNet/IP network
- recommended practices to follow for consistent %MW memory address assignment to remote device inputs and outputs

Activating the Local Slave

When a new network is created, the Unity Pro EtherNet/IP configuration tool adds a local slave node and—by default—assigns it the device **Number** of 000. Because the local slave function is not yet activated, the local slave's inputs and outputs are not initially assigned a %MW memory address.

The following example describes the effect of activating the EtherNet/IP communication module's local slave function after another remote device has already been configured and added to the network.

The sample EtherNet/IP network consists only of two nodes:

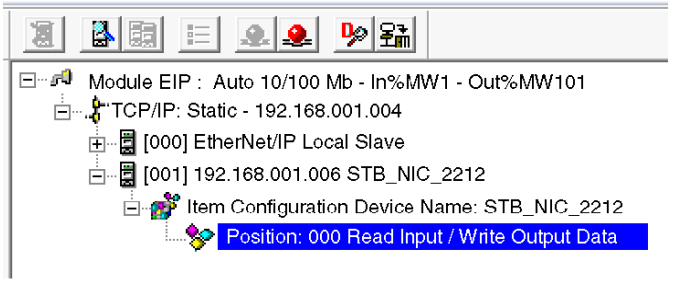
- the de-activated local slave at position 000
- a single, activated remote device at position 000

The sample EtherNet/IP network has been configured as follows:

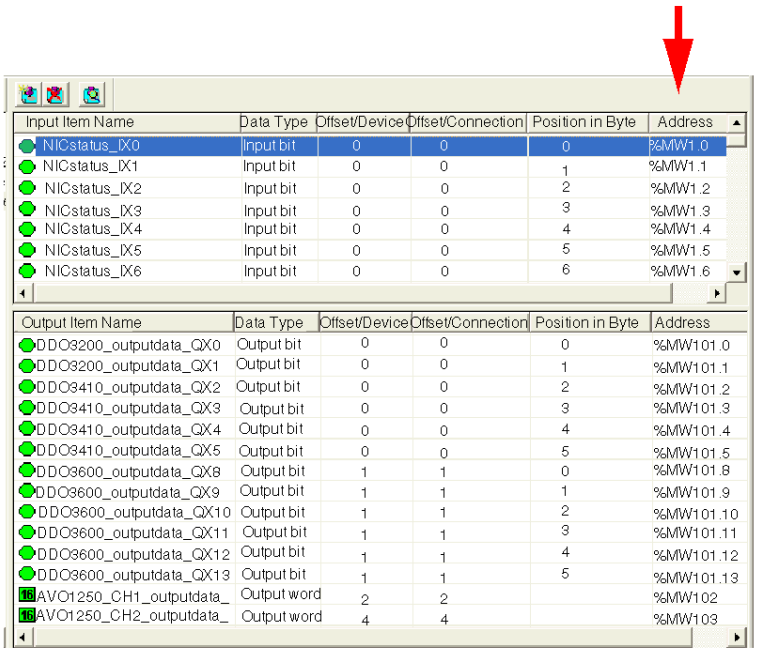
- Total EtherNet/IP network inputs and outputs are set in the **Configuration** page of the EtherNet/IP communication module in Unity Pro:
  - 100 input words are reserved, beginning at %MW01
  - 100 output words are reserved, beginning at %MW101

- Local Slave inputs and outputs:
  - 130 input bytes (65 words) are reserved
  - 130 output bytes (65 words) are reserved
- Remote device inputs and outputs:
  - 40 input bytes (20 words) are reserved
  - 40 output bytes (20 words) are reserved

The **Devices** window of the Unity Pro EtherNet/IP configuration tool displays the network, as follows:



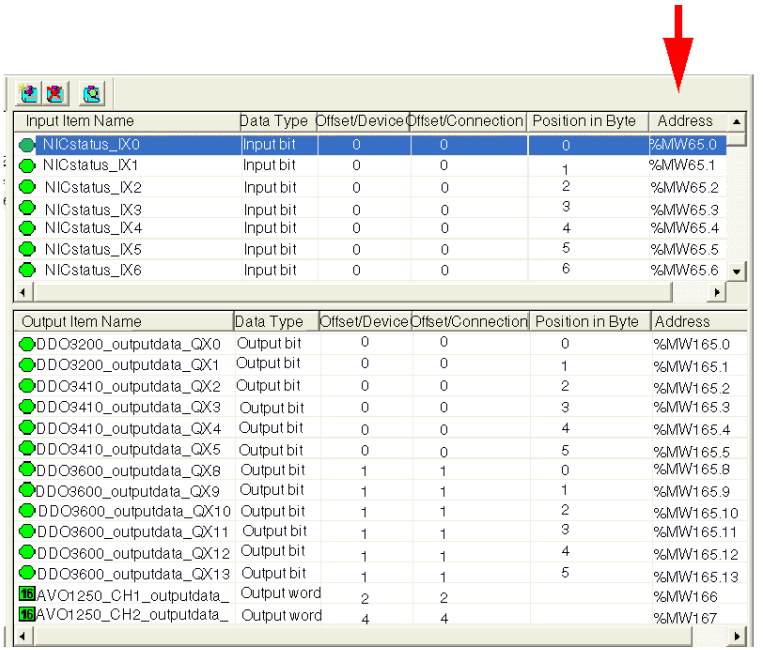
When you select the I/O Items node for the remote device, as indicated above, you display its previously configured input and output items—revealing their %MW memory address assignments:



Input Item Name	Data Type	Offset/Device	Offset/Connection	Position in Byte	Address
NICstatus_IX0	Input bit	0	0	0	%MW1.0
NICstatus_IX1	Input bit	0	0	1	%MW1.1
NICstatus_IX2	Input bit	0	0	2	%MW1.2
NICstatus_IX3	Input bit	0	0	3	%MW1.3
NICstatus_IX4	Input bit	0	0	4	%MW1.4
NICstatus_IX5	Input bit	0	0	5	%MW1.5
NICstatus_IX6	Input bit	0	0	6	%MW1.6

Output Item Name	Data Type	Offset/Device	Offset/Connection	Position in Byte	Address
DDO3200_outputdata_QX0	Output bit	0	0	0	%MW101.0
DDO3200_outputdata_QX1	Output bit	0	0	1	%MW101.1
DDO3410_outputdata_QX2	Output bit	0	0	2	%MW101.2
DDO3410_outputdata_QX3	Output bit	0	0	3	%MW101.3
DDO3410_outputdata_QX4	Output bit	0	0	4	%MW101.4
DDO3410_outputdata_QX5	Output bit	0	0	5	%MW101.5
DDO3600_outputdata_QX8	Output bit	1	1	0	%MW101.8
DDO3600_outputdata_QX9	Output bit	1	1	1	%MW101.9
DDO3600_outputdata_QX10	Output bit	1	1	2	%MW101.10
DDO3600_outputdata_QX11	Output bit	1	1	3	%MW101.11
DDO3600_outputdata_QX12	Output bit	1	1	4	%MW101.12
DDO3600_outputdata_QX13	Output bit	1	1	5	%MW101.13
AVO1250_CH1_outputdata_	Output word	2	2		%MW102
AVO1250_CH2_outputdata_	Output word	4	4		%MW103

If you next activate the local slave function, by selecting **Active Configuration** in the **General** page of its **Properties** window, then re-open the I/O items node for the remote device, you will see that the %MW memory address assignments have changed—because they now are located behind the local slave's inputs and outputs:



Input Item Name	Data Type	Offset/Device	Offset/Connection	Position in Byte	Address
NICstatus_IX0	Input bit	0	0	0	%MW65.0
NICstatus_IX1	Input bit	0	0	1	%MW65.1
NICstatus_IX2	Input bit	0	0	2	%MW65.2
NICstatus_IX3	Input bit	0	0	3	%MW65.3
NICstatus_IX4	Input bit	0	0	4	%MW65.4
NICstatus_IX5	Input bit	0	0	5	%MW65.5
NICstatus_IX6	Input bit	0	0	6	%MW65.6

Output Item Name	Data Type	Offset/Device	Offset/Connection	Position in Byte	Address
DDO3200_outputdata_QX0	Output bit	0	0	0	%MW165.0
DDO3200_outputdata_QX1	Output bit	0	0	1	%MW165.1
DDO3410_outputdata_QX2	Output bit	0	0	2	%MW165.2
DDO3410_outputdata_QX3	Output bit	0	0	3	%MW165.3
DDO3410_outputdata_QX4	Output bit	0	0	4	%MW165.4
DDO3410_outputdata_QX5	Output bit	0	0	5	%MW165.5
DDO3600_outputdata_QX8	Output bit	1	1	0	%MW165.8
DDO3600_outputdata_QX9	Output bit	1	1	1	%MW165.9
DDO3600_outputdata_QX10	Output bit	1	1	2	%MW165.10
DDO3600_outputdata_QX11	Output bit	1	1	3	%MW165.11
DDO3600_outputdata_QX12	Output bit	1	1	4	%MW165.12
DDO3600_outputdata_QX13	Output bit	1	1	5	%MW165.13
AVO1250_CH1_outputdata_	Output word	2	2		%MW166
AVO1250_CH2_outputdata_	Output word	4	4		%MW167

This shift of %MW input and output memory address assignments occurs because the assignment of a remote device's—or a local slave's—I/O to a specific %MW memory address depends upon the node's relative position among active nodes in the EtherNet/IP network.

You can avoid this shift in input and output %MW memory addresses. When you activate the local slave function, be sure to change the local slave's device **Number** from the default value of 000 to a value larger than the device number of the last device in the network.

In this example, setting the local slave's device **Number** to **002** would preserve the remote device's original %MW input and output memory address assignments.

Recommended Practices

To avoid the problem of shifting input and output %MW memory address assignments, consider the following recommended practices when developing your application:

- As described above, when activating the local slave function of an EtherNet/IP communication module, change the local slave device **Number** from its default value of 000 to a value larger than the device number for the last device in your network.
- When adding a new remote device to your EtherNet/IP network, always add it to the end of the device list and assign it a device **Number** larger than any other device number on your network.
- When configuring function blocks in Unity Pro, do not directly assign input and output pins to a specific %MW memory address. Instead, assign input and output pins to the derived data types and variables automatically created by Unity Pro.

3.2 Adding and Configuring Remote Devices

Overview

This section describes how to:

- add a generic device to your EtherNet/IP network
- configure properties for the generic device
- save, transfer and re-use Unity Pro project files that include EtherNet/IP module settings

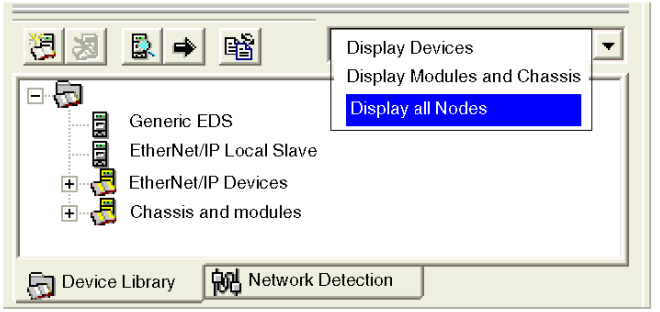
What Is in This Section?

This section contains the following topics:

Device Library

Overview

The Unity Pro EtherNet/IP configuration tool includes a **Device Library**, located in the lower left part of the configuration tool's main window. The **Device Library** is a repository of both generic and device-specific EDS files. Each EDS file defines a device, chassis, or module that you can add to your EtherNet/IP network configuration.



Both the Generic EDS node and the EtherNet/IP Local Slave node describe generic devices and cannot be deleted.

Click on the ⊞ icon to expand the device list and display the items of the selected type.

Functions

Use the **Device Library's** toolbar controls to perform the following tasks:

Function	Icon	Description
Add an EDS File		Opens the Add an EDS File wizard *(see page 73)*, which steps you through the process of adding a new EDS file to the **Device Library**.
Delete a device from the Device Library list		Deletes the selected device, chassis, or module from the Device Library list, but retains the associated EDS File in your PC's EDS File folder. You can use the Add an EDS File button  to restore the deleted device to the list. **Notes:** • Do not delete a device that has been added to your EtherNet/IP network. • You can delete only device-specific devices; you cannot delete a generic device.
Display device properties		Opens the properties window for the selected device. In the properties window, click the **View or Print EDS File...** button to display the EDS File in a text file window. In the text file window, select **File →Print** to print the contents of the EDS file.
Insert a device into your EtherNet/IP configuration		Inserts the selected device to the last position in your EtherNet/IP design. **Note:** You cannot manually insert a chassis or module into the configuration. These are added during the configuration of modular devices.
Sort the Device Library list		Opens the **Sort Device Library** window, where you can select a sort order for the devices, chassis, and modules displayed in the **Device Library**.
Filter the Device Library list	List	Click inside the drop-down list to display and select one of the following filtering options: • **Display Devices**: displays only devices—module and chassis entries are filtered out • **Display Modules and Chassis**: displays both chassis and for modules—devices are filtered out • **Display all Nodes**: displays devices, modules and chassis.

Add an EDS File to the Device Library

Overview

The Unity Pro EtherNet/IP configuration tool includes an **EDS Management** wizard that you can use to add one or more EDS files to the **Device Library**. The wizard presents a series of instruction screens that:

- simplify the process of adding EDS files to the **Device Library**, and
- provide a redundancy check in case you attempt to add duplicate EDS files to the **Device Library**

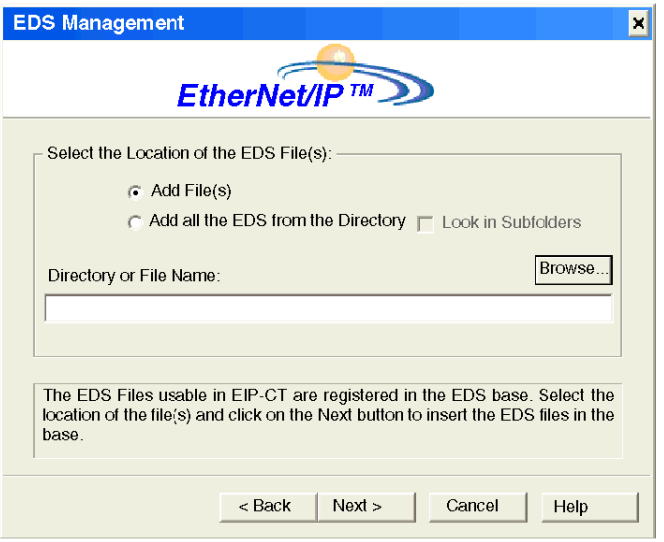
Select **Devices** →**Options...** to open the **Display Options** window, where you can enable/disable messages indicating the EDS file you are adding is a duplicate—or a different version—of an existing EDS file.

NOTE: The Unity Pro EtherNet/IP configuration tool contains a library of EDS files registered with the ODVA. This library includes EDS files for products not manufactured or sold by Schneider Electric. The non-Schneider Electric EDS files are identified in the Unity Pro EtherNet/IP Configuration Tool library. Please contact the identified device's manufacturer for inquiries regarding the corresponding non-Schneider Electric EDS files.

Adding EDS Files

To add one or more EDS files to the **Device Library**:

Step	Action
1	Do one of the following: • in the **Device Library**, click the Add button  , or • select **Library** →**Add** Page 1 of the wizard opens.

Step	Action
2	Click **Next**. Page 2 of the wizard opens: 
3	In the **Select the Location of the EDS File(s)** section, select either: ● **Add File(s)**, to add one or more EDS files you will individually select, or ● **Add all the EDS Files from the Directory**, to add all files from a folder you will select. ● Select **Look in Subfolders** to also add EDS files in subfolders beneath the folder you select
4	Click the **Browse** button. The **Open** dialog opens.
5	Use the **Open** dialog to navigate to and select: ● one or more EDS files, or ● a folder containing EDS files
6	After you have made your selection9s), click **Open**. The dialog closes and your selection appears in the **Directory or File Name** field.
7	Click **Next**. The wizard compares the selected EDS files against existing files in the **Device Library**.
8	(Conditional) If one or more selected EDS files are duplicates and if notice of redundant files is enabled in the **Display Options** dialog, a **File Already Exists** message displays. **Close** the message.

Step	Action
9	Page 3 of the wizard opens indicating the Status of each device you attempted to add: ● a green check mark indicates the EDS file can be added ● a blue informational icon indicates a redundant file ● a red check mark indicates an invalid EDS file (Optional) Select a file in the list, then click **View Selected File** to open it.
10	Click **Next** to add the non-duplicate files. Page 4 of the wizard opens, indicating the action is complete.
11	Click **Finish** to close the wizard.

Adding A Remote Device

Overview

The Device Library consists of two types of entries:

Entry	Defined by
generic	A device without an associated EDS File. In the Device Library, generic devices include: ● Generic EDS ● EtherNet/IP Local Slave
EDS File specific	A device, module, or chassis defined by a unique vendor-created EDS File. In the Device Library, these devices appear beneath the branches: ● EtherNet/IP Devices ● Chassis and modules

You can add both generic devices or devices with a specific EDS File to your EtherNet/IP network.

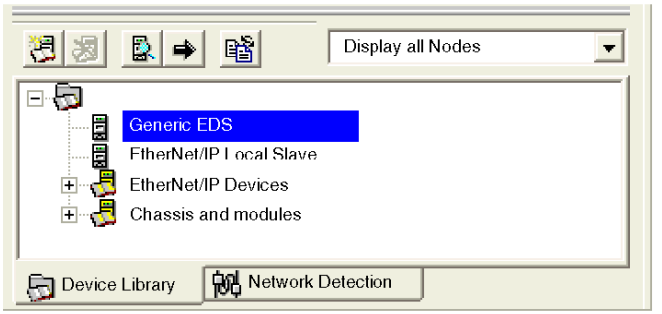
When you add:

- a device that is associated with a specific EDS File, the Unity Pro EtherNet/IP configuration tool recognizes the device and automatically performs much of the device configuration for you
- a generic device, you need to manually perform all device configuration

In the following example, a generic device is added to an EtherNet/IP network.

Adding a Generic Remote Device

To add a generic remote device to your EtherNet/IP network, follow these steps:

Step	Action
1	In the **Device Library**, select Generic EDS (see below): 

Step	Action
2	Click the Insert ➡ button. Two things occur simultaneously: ● a new generic device is added to the end of the EtherNet/IP network configuration, and ● the Generic EDS properties window opens for editing.
3	Refer to the topic Configuring a Generic Remote Device *(see page 78)* for additional instructions on configuring the generic device.

Configuring Remote Device Properties

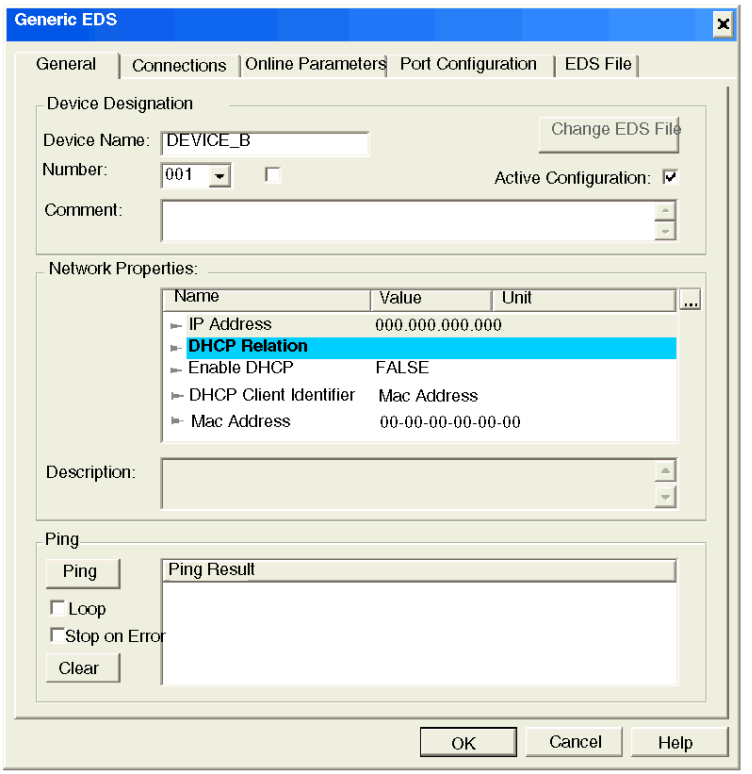
Overview

When a generic device is added to an EtherNet/IP network, the Unity Pro EtherNet/IP configuration tool automatically opens its properties window for immediate configuration. When operating offline, the properties window consists of the following 5 pages. Only the first two of these pages need to be configured:

In this page...	Do the following...
General	Enter configuration settings, as described below.
Connections	Enter configuration settings, as described below.
Online Parameters	Not accessible offline. No configuration required.
Port Configuration	Not accessible offline. No configuration required.
EDS File	(Read-only page - no configuration required)

Configuring the General Page

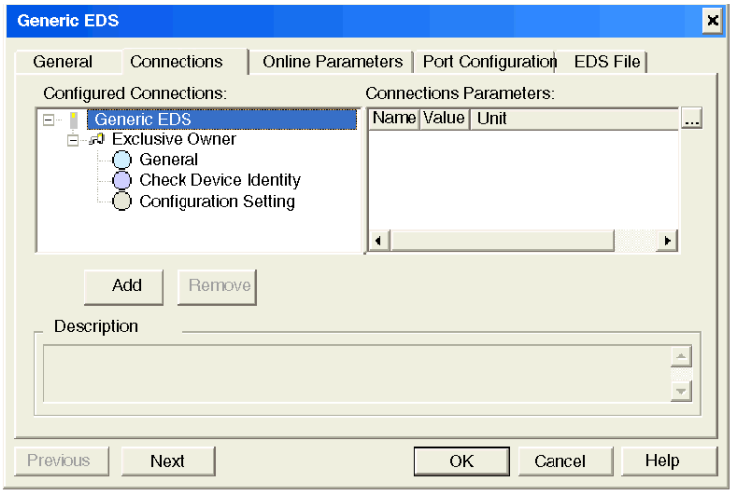
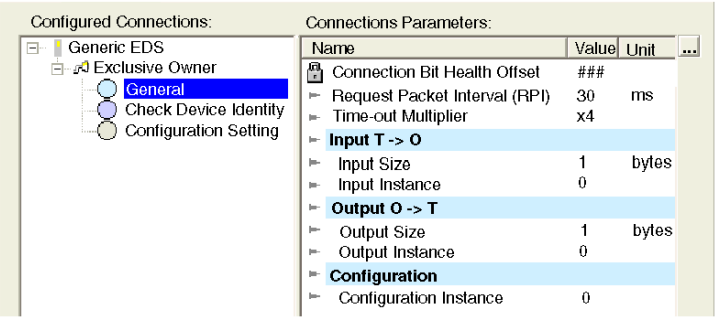
To configure the **General** page:

Step	Action
1	Click on the **General** page: 

Step	Action	
2	In the **General** page, edit the following settings:	
	Device Name	The label for the remote device in the EtherNet/IP device list. Either: ● type in a unique name using letters, numbers and the underscore character (_), or ● accept the auto-generated name (DEVICE_N}
	Number	The relative position in the EtherNet/IP device list. Either: ● accept the default (i.e. the next available number), or ● select a different number from the drop-down list
	Link Parameters	Select this setting to: ● lock the **IP Address** setting, and ● set the last octet of the **IP Address** equal to the value selected in the **Number** parameter De-select this setting to unlock the **IP Address** setting.
	Active Configuration	Select this setting to include this remote device in EtherNet/IP network communications. De-select this setting to exclude this device from network communications, but save the device's configuration settings.
	IP Address	The IP Address of this remote device. This setting is: ● editable, when the **Link Parameters** field is de-selected ● locked, when the **Link Parameters** field is selected By default: ● the first 3 octet values equal the first 3 octet values of the EtherNet/IP module's IP address ● when the **Link Parameters** field is selected, the last octet value equals the value selected in the **Number** parameter
	Enable DHCP	TRUE activates the DHCP client in this remote device. On startup, this device requests its IP address from a DHCP server. **Note**: the EtherNet/IP module can be configured to act as a DHCP server.
	DHCP Client Identifier	If the DHCP client is enabled, select the identifier the DHCP server will use to recognize this remote device: ● **MAC Address** ● **Device Name**
	Mac Address/Device Name	Type in the value of the DHCP client identifier. **Note:** The **Device Name** referenced here is not the same as the Device Name described in the first row of this table.

Configuring the Connections Page

To configure the Connections page:

Step	Action
1	Click on the **Connections** page: 
2	In the **Configured Connections** list, click on **General** to display the general connection settings in the **Connection Parameters** list, shown below: 

Step	Action	
3	In the **Connections** page, edit the following general connection settings:	
	Connection Health Bit Offset	(read-only)
	Request Packet Interval (RPI)	The refresh period for this I/O connection. Value range: 2...65535 ms Default = 30 ms
	Time-out Multiplier	The value, multiplied against the RPI rate, which triggers an inactivity timeout. Value list: 4, 8, 16, 32, 64, 128, 256, 512 Default: 4
	Input Size (in bytes)	The number of bytes reserved for input data, in bytes. Value range: 1...509 Default: 1
	Input Instance	The instance identifier for inputs: **101**.
	Output Size (in bytes)	The number of bytes reserved for output data, in bytes. Value range: 1...505 Default: 1
	Output Instance	The instance identifier for outputs: **102**.
	Configuration Instance	The instance identifier for configuration data: **103**.
	Note: The **Input Size** and **Output Size** parameter settings are determined by the size—in bytes—of the input data and output data sections of your specific application.	
4	Click **OK** to save your settings and close the **Properties** window. The next step is to configure I/O settings. For an example of I/O configuration for a generic remote device, see how the following I/O items were configured: ● discrete input items *(see page 101)* ● discrete output items *(see page 104)* ● numeric input items *(see page 107)* ● numeric output items *(see page 110)*	

Managing Project Files

Overview

Managing Unity Pro project files that contain EtherNet/IP module settings includes:

- saving project files as either:
  - Unity Pro Archived Application Files (*.STA)
  - Unity Pro project files (*.STU)
- opening saved project files
- transferring files

NOTE: To transfer Unity Pro project files, follow the steps set forth below.

Do not use the following Unity Pro commands to transfer a Unity Pro project file that contains EtherNet/IP settings:

- project transfer command: **PLC →Transfer Project from PLC**
- export project command: **File →Export Project...**

Creating Unity Pro Archive (*.STA) Files

Unity Pro project files, containing EtherNet/IP module settings, can be transferred within the Unity Pro application only as Unity Pro Archived Application Files (*.STA). To save a Unity Pro project file as a Unity Pro Archived Application File (*.STA) suitable for transfer and reuse, follow these steps:

Step	Action
1	Build the Unity Pro project. Select: **Build →Rebuild All Project**.
2	Download the rebuilt Unity Pro project file to the PLC. Select: **PLC →Transfer Project to PLC**. The taskbar should indicate **EQUAL**.
3	Go offline. Select: **PLC →Disconnect**.
4	Select **File →Save Archive...** The **Save Archive** window opens.
5	In the **Save Archive** window: • type a **File name** • navigate to a location to store the archived project file • click **Save**. Unity Pro creates a Unity Pro Archived Application File (*.STA).

Opening a Unity Pro Archive (*.STA) File

After a Unity Pro Archived Application File has been saved, you can transfer it (like any file), then re-open it in the same version of Unity Pro. To re-open an archived project file:

Step	Action
1	Select **File** →**Open**. The **Open** dialog opens.
2	In the **Open** dialog, select **Unity Pro Archived Application Files (*.STA)** as the **Files of type**.
3	In the **Look in** drop down box, navigate to the location of the archived Unity Pro archive file that you want to open.
4	Select the file and click **Open**. Unity Pro opens the archived Unity Pro project file.

Transferring Unity Pro Project (.STU) Files

You can copy, paste, and transfer a Unity Pro project (*.STU) file as you would any file, using the tools and commands available in Windows Explorer.

A saved Unity Pro project (*.STU) file can be re-opened only by the same version of Unity Pro software that saved it.

3.3 Configuring the STB NIC 2212

Overview

This section presents a sample configuration of an STB NIC 2212 EtherNet/IP network interface module, and adds it to a Unity Pro project.

NOTE: The instructions in this chapter describe a single, specific device configuration example. Refer to the Unity Pro EtherNet/IP configuration tool help file for additional information about alternative configuration choices.

The following example extends the sample configuration of the EtherNet/IP communications network—described in the previous chapter—where you:

- created a project
- added a power supply module, CPU and EtherNet/IP communication module to the project
- configured the EtherNet/IP communication module

What Is in This Section?

This section contains the following topics:

Setting Up Your Network

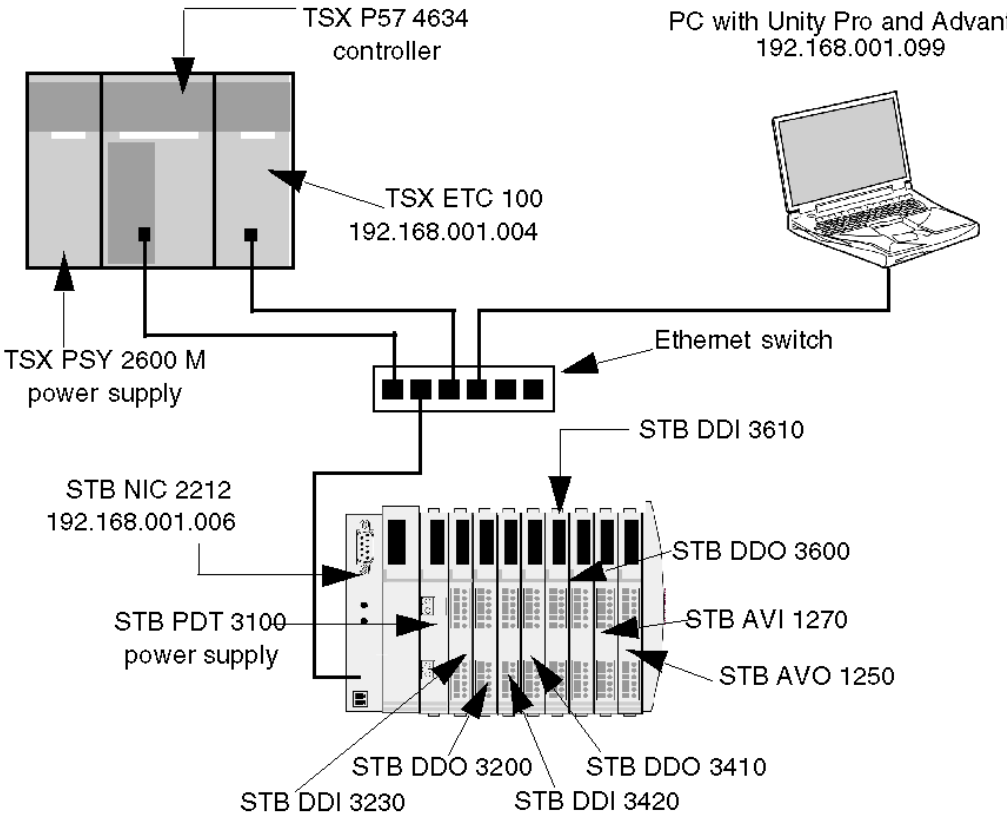
Overview

This sample network includes the following hardware and software:

- a controller rack with:
  - TSX PSY 2600 M, 115/230 VAC power supply
  - TSX P57 4634, 14A controller
  - TSX ETC 100, 10/100 Base-T EtherNet/IP communication module
- a remote STB Advantys island with:
  - STB NIC 2212 EtherNet/IP network interface module
  - STB PDT 3100 power distribution module
  - STB DDI 3230 2 pt digital input module
  - STB DDO 3200 2 pt digital output module
  - STB DDI 3420 4 pt digital input module
  - STB DDO 3410 4 pt digital output module
  - STB DDI 3610 6 pt digital input module
  - STB DDO 3600 6 pt digital output module
  - STB AVI 1270 2 pt analog input module
  - STB AVO 1250 2 pt analog output module
- a PC running both Unity Pro (version 4.0 or higher) and Advantys configuration software (version 4.0 or higher)
- an Ethernet switch connected to the above EtherNet/IP devices with twisted pair Ethernet cable and RJ45 connectors (It is strongly recommended you select a managed switch that supports the IGMP protocol.)

Network Topology

The network example topology looks like this:



To re-create this example, be sure to:

- use the IP addresses for your own configuration's:
  - PC
  - TSX ETC 100 EtherNet/IP communication module
  - STB NIC 2212 EtherNet/IP network interface module
- check all wiring

NOTE: Unity Pro software running in the PC is used to configure the TSX P57 60 controller. In this example, the PC is indirectly wired to the CPU's Ethernet port via the Ethernet switch. Alternatively, you could bypass the switch and directly wire the PC to another one of the CPU's ports.

Automatically Detect and Add the STB NIC 2212

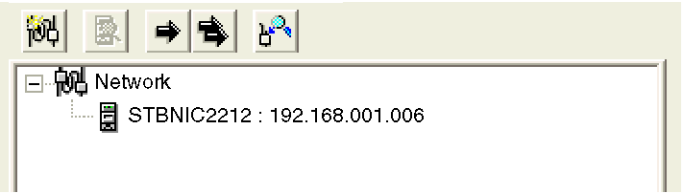
Overview

Use the Unity Pro EtherNet/IP configuration tool to automatically detect the STB NIC 2212 module, then add it to your project.

NOTE: The STB NIC 2212 module must be active online with a valid IP address before you can detect it then add it to your project. You can assign an IP address using a DHCP or BOOTP server, or use the MAC-generated (default) IP address.

Detecting and Adding Network Devices

To automatically detect the STB NIC 2212, then add it to your project, follow these steps:

Step	Action
1	Launch the configuration tool from the **Configuration** page of the EtherNet/IP communication module's **Properties** window.
2	In the configuration tool, begin on-line operations by clicking the **Go Online** button .
3	Click on the **Network Detection** tab to enable automatic network detection: 
4	Click the **Read Network Configuration** toolbar button . The configuration tool searches the network for EtherNet/IP devices, classifies them using the device EDS file, then lists the EtherNet/IP devices it detects. 
5	Select the **STB NIC 2212** in **Network Detection** window.
6	Click the **Insert in Configuration** button . The properties window opens, where you can configure the STB NIC 2212.

Configuring STB NIC 2212 Properties

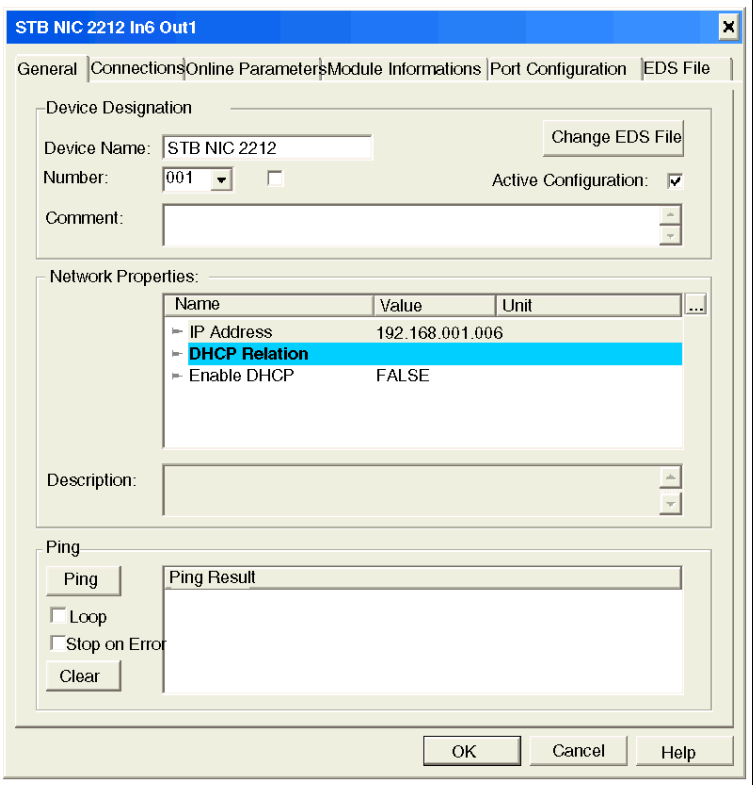
Overview

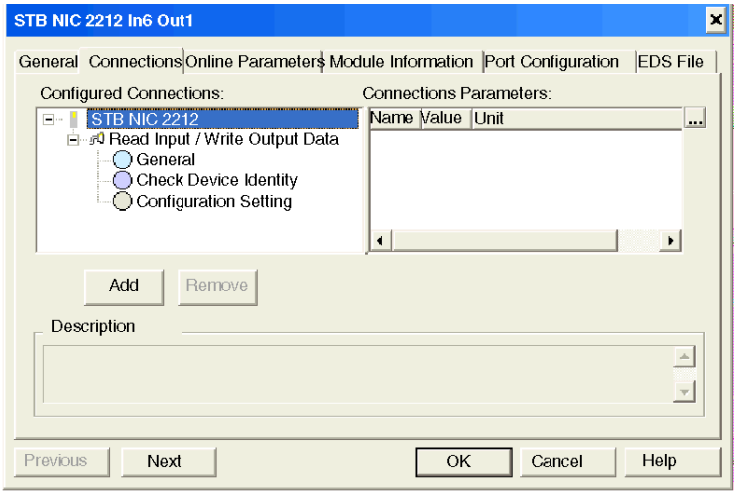
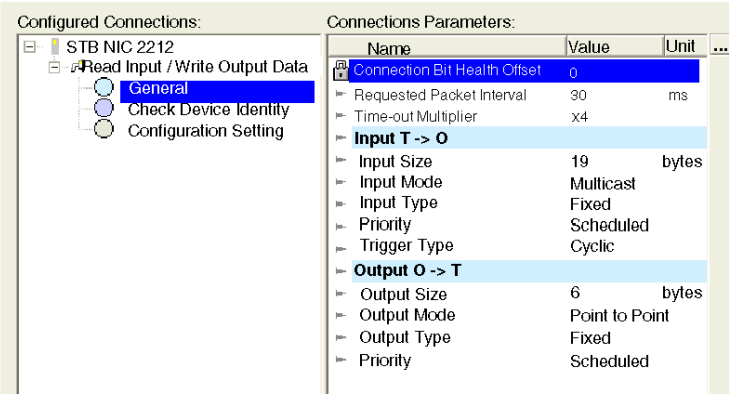
The STB NIC 2212 EtherNet/IP network interface module properties window presents the following tabbed pages. Only some of these pages need to be edited for this example:

In this page...	Do the following...
General	• input device name • configure IP address • add the device to the project configuration
Connections	• configure the requested packet interval (RPI) • specify the size and location of inputs and outputs
Online Parameters	Accept the default settings, if any.
Module Informations	(Read-only page - no configuration required)
Port Configuration	(Read-only page - no configuration required)
EDS File	(Read-only page - no configuration required)

Configuring the STB NIC 2212

The following settings are used in this sample configuration. Be sure to use settings that are appropriate for your actual application:

<table>
<tr><th>Step</th><th colspan="2">Action</th></tr>
<tr><td>1</td><td colspan="2">Click on the General page:
 
</td></tr>
<tr><td>2</td><td colspan="2">In the General page, edit the following settings:</td></tr>
<tr><td></td><td>Device Name</td><td>STB NIC 2212</td></tr>
<tr><td></td><td>Number</td><td>The relative position in the EtherNet/IP device list. For this example, select 001.</td></tr>
<tr><td></td><td>Active Configuration</td><td>Be sure this checkbox is selected.</td></tr>
<tr><td></td><td>IP Address</td><td>192.168.001.006</td></tr>
</table>

Step	Action
3	Click on the **Connections** page: 
4	In the **Configured Connections** list, click on **General** to display the general connection settings in the **Connection Parameters** list, shown below: 

<table>
<tr><th>Step</th><th colspan="2">Action</th></tr>
<tr><td rowspan="6">5</td><td colspan="2">In the Connections page, edit the following general connection settings:</td></tr>
<tr><td>Request Packet Interval</td><td>30 ms</td></tr>
<tr><td>Input Size (in bytes)</td><td>19 bytes</td></tr>
<tr><td>Input Instance</td><td>101</td></tr>
<tr><td>Output Size (in bytes)</td><td>6 bytes</td></tr>
<tr><td>Output Instance</td><td>102</td></tr>
<tr><td colspan="2">Note: The Input Size and Output Size parameter settings are determined by the size—in bytes—of the input data and output data sections of the Advantys island's Fieldbus Image.</td></tr>
<tr><td>6</td><td colspan="2">Click OK to save your settings and close the Properties window.
A node is added to the project configuration in the Devices window, below:
 

The next step is to configure I/O settings.</td></tr>
</table>

Connecting to the Advantys STB Island

Overview

In this example, you will use the Advantys configuration software running on your PC to:

- connect the Advantys configuration software to the STB NIC 2212 and the 8 I/O modules that comprise the Advantys STB island
- upload Advantys STB island configuration to the Advantys configuration software in your PC
- display a fieldbus image for the Advantys STB island showing the relative location of:
  - status information
  - input data
  - output data

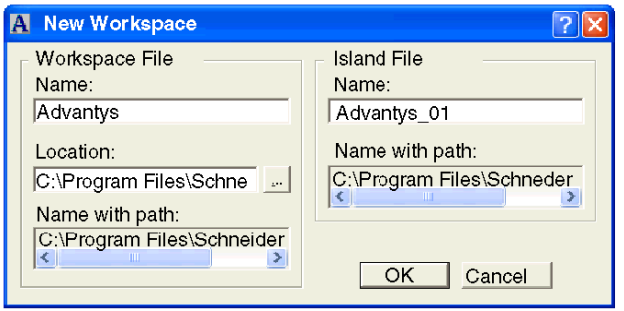
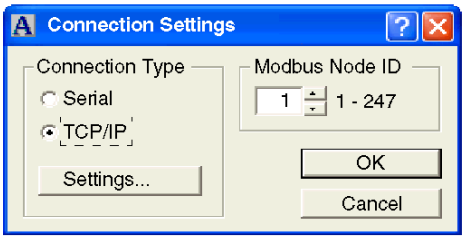
Using the data presented in the fieldbus image, you can use the Unity Pro EtherNet/IP configuration tool to create input and output items that map to specific status, input, output, and output echo data.

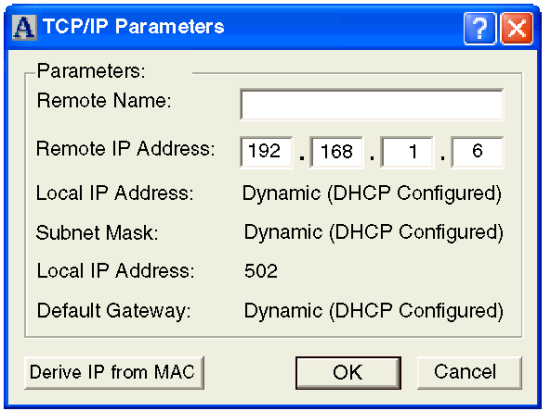
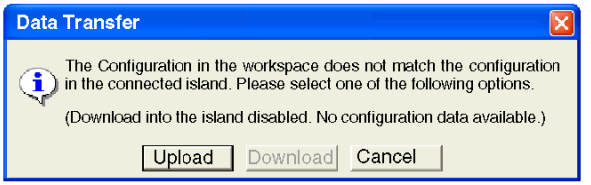
NOTE: Before proceeding with the following instructions, be sure you have auto-configured the Advantys STB island by pressing the **RST** button on the front of the STB NIC 2212 module.

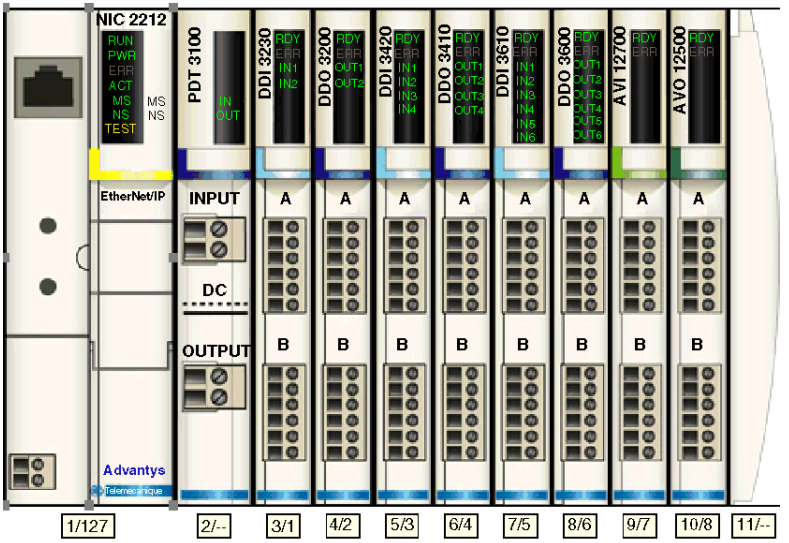
Making the Connection

To connect to the STB NIC 2212 and I/O modules using the Advantys configuration software:

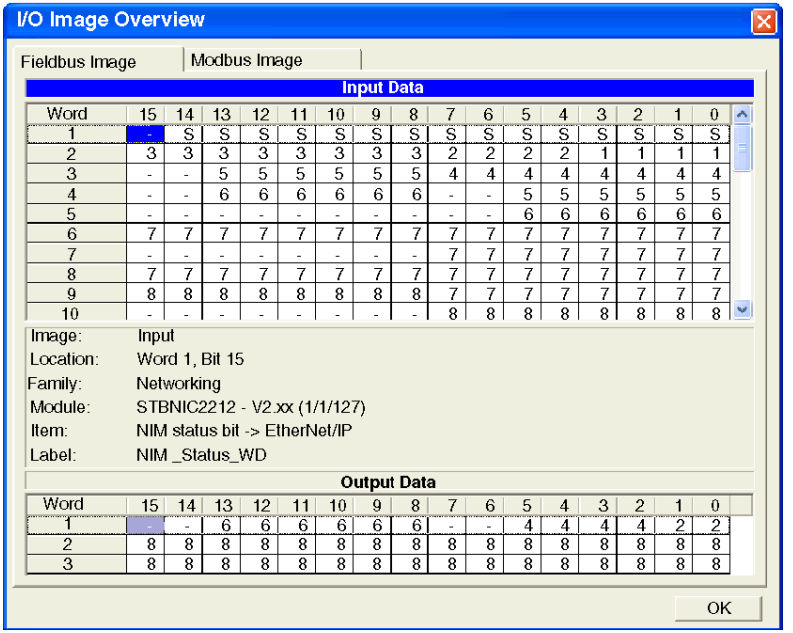
Step	Action
1	Startup the Advantys configuration software on your PC. A dialog opens displaying available project types.
2	Select **STB**. A choice of language dialog opens.
3	Select your choice of language.
4	Select **File** →**New Workspace**. The **New Workspace** window opens (below).

Step	Action
5	For this example, type in the following field values: ● for the field **Workspace File** type in **Advantys** ● for the field **Island File** type in **Advantys_01** 
6	Click **OK**. The Advantys configuration software displays an empty DIN rail in the center of the screen.
7	Select **Online** →**Connection Settings**. The **Connection Settings** window opens (below).
8	In the **Connection Settings** window, accept the **Modbus Node ID** default setting of **1**, select **TCP/IP**, and click the **Settings...** button:  The **TCP/IP Parameters** dialog opens (below).

Step	Action
9	In the **Remote IP Address** field, type in the IP address for the STB NIC 2212, in this example: **192.168.1.6**.  **Note:** Use the mouse to move between octets, and do not type in any leading zeroes. For example, do not type in 192.168.001.006.
10	Click **OK** to close the **TCP/IP Parameters** dialog, and click **OK** again to close the **Connection Settings** dialog.
11	Select **Online** →**Connect**. The **Data Transfer** dialog opens (below): 

Step	Action
12	Select **Upload** in the **Data Transfer** dialog. The island workspace is populated with island data and shows the STB NIC 2212 and all island modules (below):  **Note:** A box appears beneath each module containing one or two integers—for example 3/1. These integers serve the following purpose: • The left-side integer (3 in this example) identifies the module's physical position—left to right—among all modules in the rack. • The right-side integer (1 in this example) identifies the module's relative position—left to right—among only data producing/receiving modules. If the module is not a data producing/receiving module (e.g. a power supply, or end of segment module) no right-side integer appears.

Step	Action
13	Select **Island →I/O Image Overview**. The **I/O Image** window opens to the **Fieldbus Image** page:



Each table cell contains one of the following alpha-numeric indicators:

- **S** indicates a status bit for the STB NIC 2212 network interface module
- **an integer** identifies the relative position—from left to right—of a data producing/receiving module with input or output data in that cell. For example:
  - the STB DDI 3230 input module is the first data producing or receiving module in the rack; its data is designated by the integer 1 in bits 0 - 3 of word 2 in the **Input Data** table
  - the STB DDO 3600 output module is the sixth data producing module in the rack; its status and output echo data is designated by the integer 6 in bits 8 - 13 of word 4 and in bits 0 - 5 of word 5 in the **Input Data** table; its output data is designated by the integer 6 in bits 8 - 13 of word 1 in the **Output Data** table

Notes:

Select a cell in either the **Input Data** or **Output Data** tables to display—in the middle of the page—a description of the cell data and its source module.

Convert the size of the **Input Data** table and the **Output Data** table from words to bytes (i.e. divide by 2), then use that data as the values for the **Input Size** (19) and **Output Size** (6) parameters when configuring the remote device's general connection properties *(see page 81)*.

Configuring I/O Items

Overview

The final task in this example is to add I/O items to the configuration of the STB NIC 2212 EtherNet/IP network interface module and its 8 I/O modules. To accomplish this:

- use the Advantys configuration software to identify the relative position of each I/O module's inputs and outputs
- use the Unity Pro EtherNet/IP configuration software to create input and output items, defining each item's:
  - name
  - data type
- identity the address assigned to each new input and output item using the Unity Pro EtherNet/IP configuration software

I/O Item Types and Sizes

The goal is to create a collection of input items and output items that equal the input size and output size specified in the STB NIC 2212 Connection properties page. In this example, items need to be created for:

- 19 bytes of inputs
- 6 bytes of outputs

The Unity Pro EtherNet/IP configuration tool provides great flexibility in creating input and output items. You can create input and output items in groups of 1 or more single bits, 8-bit bytes, 16-bit words, 32-bit dwords, or 32-bit IEEE floating values. The number of items you create depends upon the data type and size of each item.

In the sample project, the following items were created:

- discrete bits for digital inputs and outputs
- 8-bit bytes or 16-bit words for analog inputs and outputs

Mapping Input and Output Items

Use the **Fieldbus Image** page of the **I/O Image Overview** window in the Advantys configuration software to identify the number and type of I/O items you need to create, as follows:

Step	Action
1	In the Advantys configuration software, select **Island** →**I/O Image Overview**. The **I/O Image** window opens to the **Fieldbus Image** page.
2	Select the first cell (word 1, cell 0) in the **Input Data** table to display—in the middle of the page—a description of the cell data and its source module.

Step	Action
3	Make a note of the word, bit(s), module and item information for that cell.
4	Repeat steps 2 and 3 for each cell containing either an S or an integer.

NOTE: The **Fieldbus Image** presents input and output data in the form of 16-bit words (starting with word 1). You need to rearrange this data for the Unity Pro EtherNet/IP configuration tool, which presents the same data in the form of 8-bit bytes (starting with byte 0).

This process yields the following tables of input and output data:

Input Data:

Advantys Fieldbus Image		Unity Pro EIP Items		STB Module	Description
Word	Bit(s)	Byte	Bit(s)		
1	0-15	0	0-7	NIC 2212	NIC status
		1	0-7		
2	0-1	2	0-1	DDI 3230	input data
	2-3		2-3	DDI 3230	input status
	4-5		4-5	DDO 3200	output data echo
	6-7		6-7	DDO 3200	output status
	8-11	3	0-3	DDI 3420	input data
	12-15		4-7	DDI 3420	input status
3	0-3	4	0-3	DDO 3410	output data echo
	4-7		4-7	DDO 3410	output status
	8-13	5	0-5	DDI 3610	input data
	14-15		6-7	NA	not used
4	0-5	6	0-5	DDI 3610	input status
	6-7		6-7	NA	not used
	8-13	7	0-5	DDO 3600	output data echo
	14-15		6-7	NA	not used
5	0-5	8	0-5	DDO 3600	output status
	6-15	8	6-7	NA	not used
		9	0-7		
6	0-15	10	0-7	AVI 1270	input data ch 1
		11	0-7		
7	0-7	12	0-7	AVI 1270	input status ch 1
	8-15	13	0-7	NA	not used

Advantys Fieldbus Image		Unity Pro EIP Items		STB Module	Description
Word	**Bit(s)**	**Byte**	**Bit(s)**		
8	0-15	14	0-7	AVI 1270	input data ch 2
		15	0-7		
9	0-7	16	0-7	AVI 1270	input status ch 2
	8-15	17	0-7	AVO 1250	output status ch 1
10	0-7	18	0-7	AVO 1250	output status ch 2
	8-15	NA	NA	NA	not used

Output Data:

Advantys Fieldbus Image		Unity Pro EIP Items		Module	Description
Word	**Bit(s)**	**Byte**	**Bit(s)**		
1	0-1	0	0-1	DDO 3200	output data
	2-5		2-5	DDO 3410	output data
	6-7		6-7	NA	not used
	8-13	1	0-5	DDO 3600	output data
	14-15		6-7	NA	not used
2	0-15	2	0-7	AVO 1250	output data ch 1
		3	0-7		
3	0-15	4	0-7	AVO 1250	output data ch 2
		5	0-7		

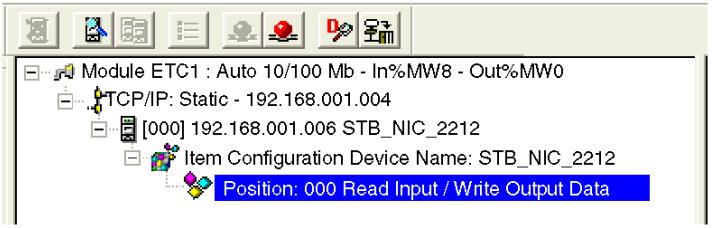
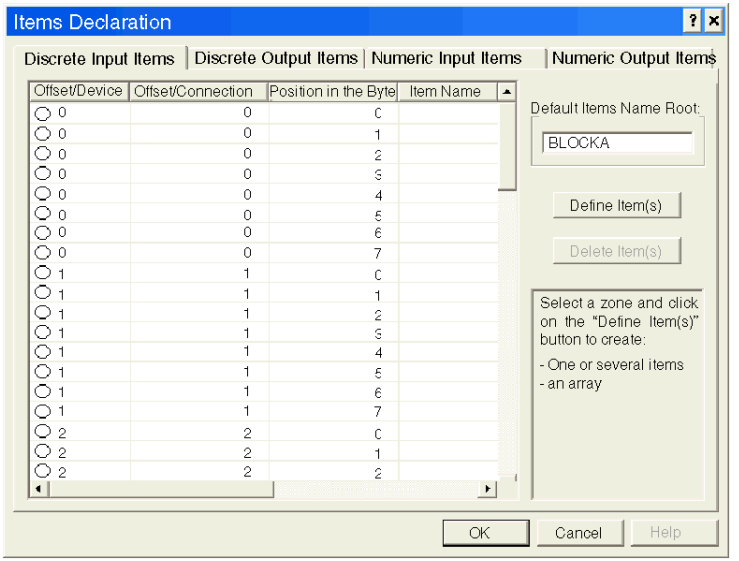
For this example, you need to create input items for the 19 input bytes and output items for the 6 output bytes using the Unity Pro EtherNet/IP configuration tool. These input and output items include:

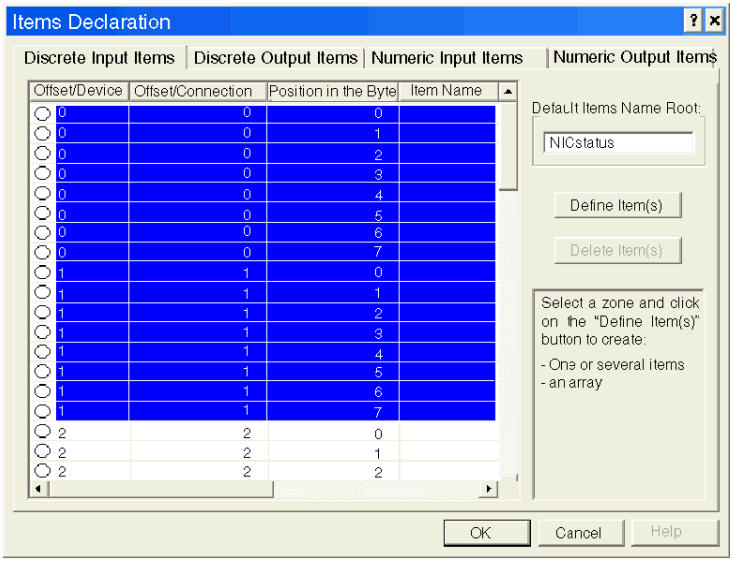
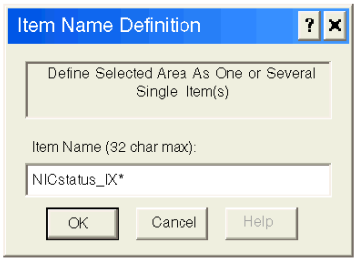
- discrete input and output items—made up of 1 or more bits—for the digital I/O modules, and
- numeric input and output items—made up of either an 8-bit byte or a 16-bit word—for the analog I/O modules

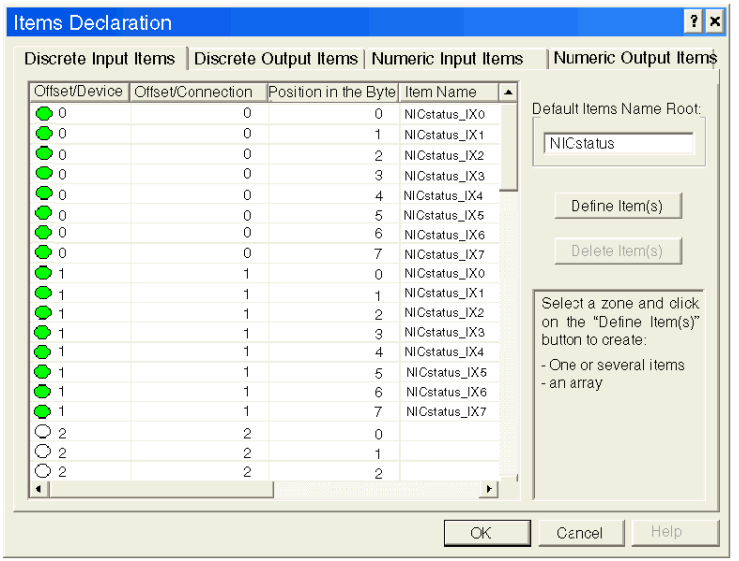
The following examples show you how to create each kind of item.

Creating Discrete Input Items

To create discrete input items for the STB NIC 2212 example, beginning with 16 discrete inputs for NIC status:

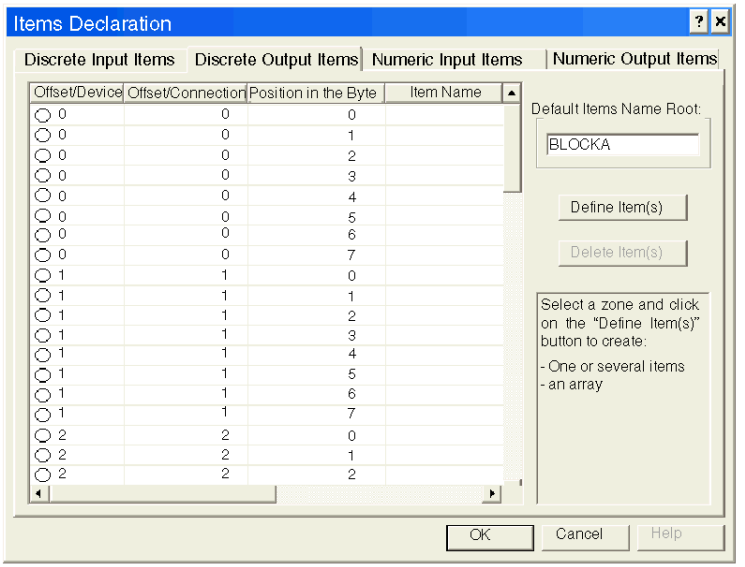
Step	Action
1	In the **Devices** window of the Unity Pro EtherNet/IP configuration tool, navigate to and select the connection item at Position 000, as shown below: 
2	Select **Devices** →**Properties**. The **Items Declaration** window opens: 
3	In the **Default Items Name Root** input box type: **NICstatus**.

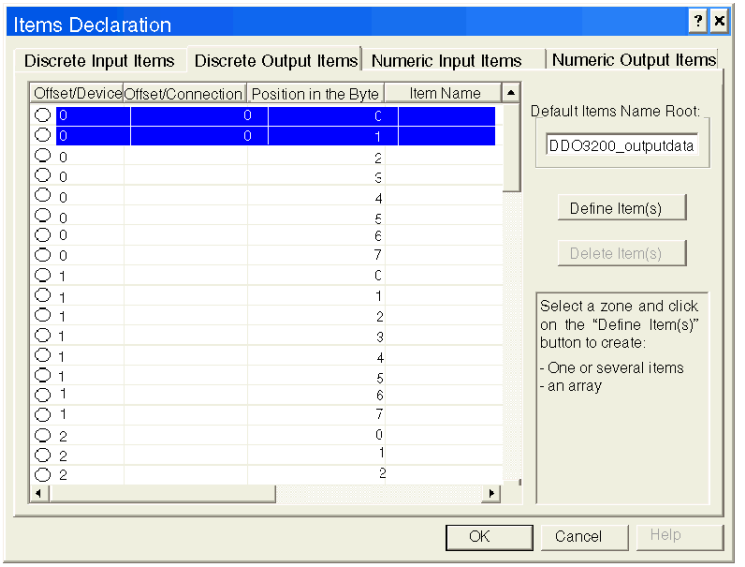
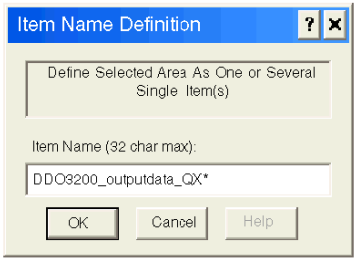
Step	Action
4	In the **Items List**, select the rows that correspond to bits 0-7 in bytes 0 and 1—i.e., the first 16 rows: 
5	Click the **Define Item(s)** button.The **Item Name Definition** dialog opens:  **Note:** The asterisk (*) indicates a series of discrete items with the same root name will be created.

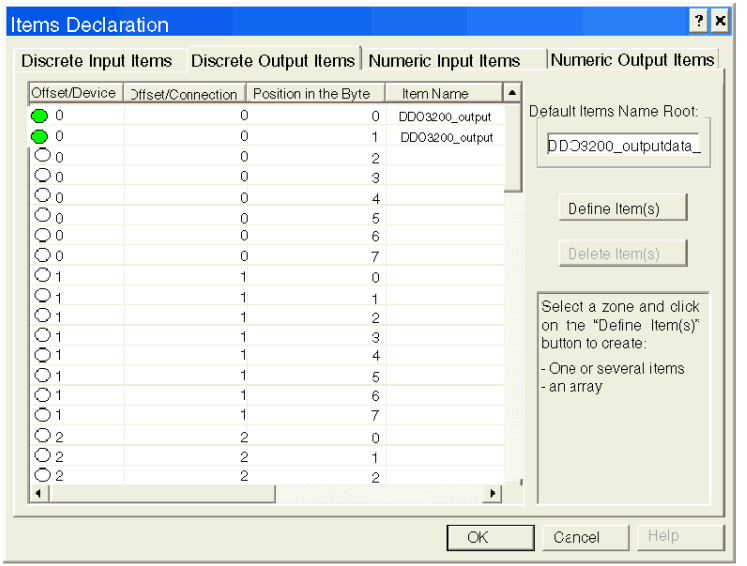
Step	Action
6	Accept the default **Item Name** and click **OK**. 16 discrete input items are created: 
7	Repeat steps 3 - 6 for each group of discrete input items you need to create. In this example, that includes items for each of the following groups: ● Byte: 2, Bits: 0-1, Default Items Name Root: DDI3230_inputdata ● Byte: 2, Bits: 2-3, Default Items Name Root: DDI3230_inputstatus ● Byte: 2, Bits: 4-5, Default Items Name Root: DDO3200_outputdataecho ● Byte: 2, Bits: 6-7, Default Items Name Root: DDO3200_outputstatus ● Byte: 3, Bits: 0-3, Default Items Name Root: DDI3420_inputdata ● Byte: 3, Bits: 4-7, Default Items Name Root: DDI3420_inputstatus ● Byte: 4, Bits: 0-3, Default Items Name Root: DDO3410_outputdataecho ● Byte: 4, Bits: 4-7, Default Items Name Root: DDO3410_outputstatus ● Byte: 5, Bits: 0-5, Default Items Name Root: DDI3610_inputdata ● Byte: 6, Bits: 0-5, Default Items Name Root: DDI3610_inputstatus ● Byte: 7, Bits: 0-5, Default Items Name Root: DDO3600_outputdataecho ● Byte: 8, Bits: 0-5, Default Items Name Root: DDO3600_outputstatus
8	Click on the **Discrete Output Items** tab to open that page.

Creating Discrete Output Items

To create discrete output items for the STB NIC 2212 example, beginning with 2 discrete outputs for the STB DDO3200 module:

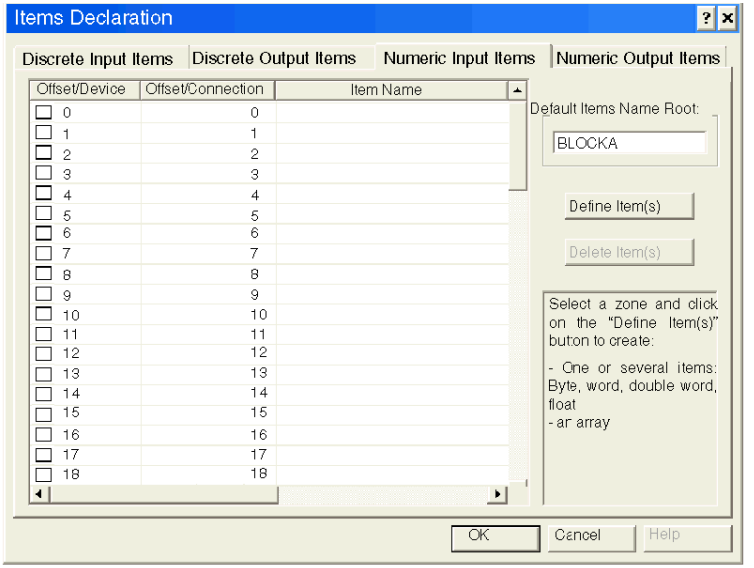
Step	Action
1	The discrete output items page looks like this:  **Note:** Again, both the **Offset/Device** and **Offset/Connection** columns represent the byte address of the discrete output, while the **Position in the Byte** column indicates the bit position of the discrete output item.
2	In the **Default Items Name Root** input box type: **DDO3200_outputdata**.

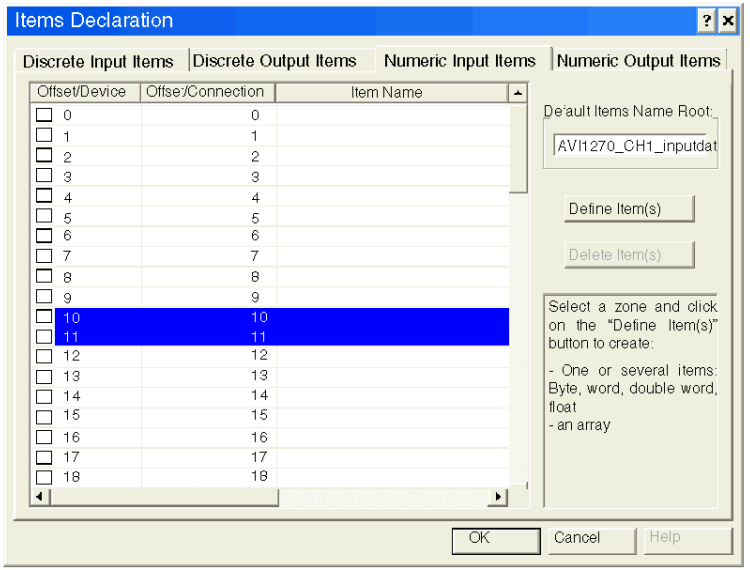
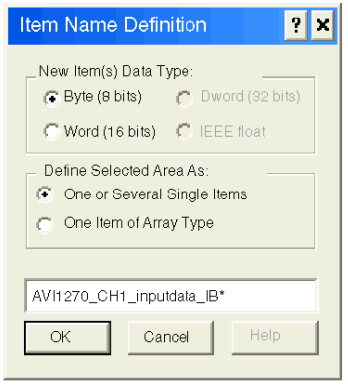
Step	Action
3	In the Items List, select the rows that correspond to bits 0-1 in byte 0—i.e., the first 2 rows: 
4	Click the **Define Item(s)** button.The **Item Name Definitio**n dialog opens:  **Note:** The asterisk (*) indicates a series of discrete items with the same root name will be created.

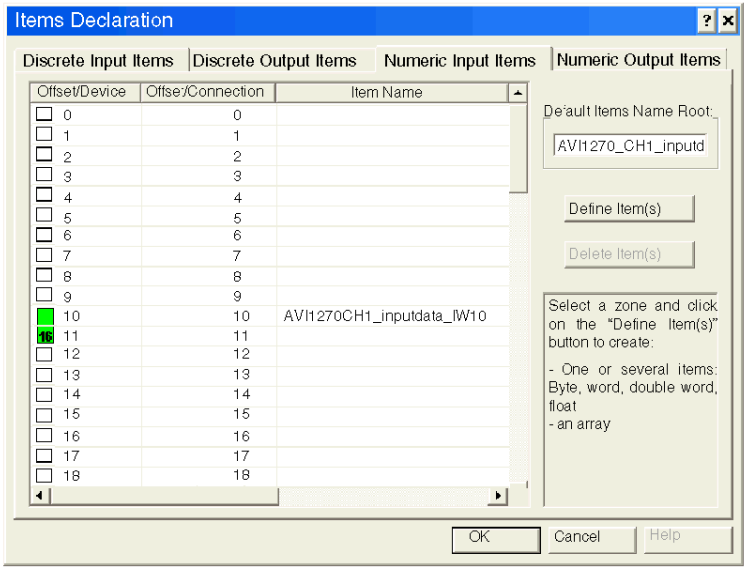
Step	Action
5	Accept the default **Item Name** and click **OK**. 2 discrete input items are created: 
6	Repeat steps 2 - 5 for each group of discrete output items you need to create. In this example, that inlcudes items for each of the following groups: ● Byte: 0, Bits: 2-5, Default Items Name Root: DDO3410_outputdata ● Byte: 1, Bits: 0-5, Default Items Name Root: DDO3600_outputdata
7	Click on the **Numeric Input Items** tab to open that page.

Creating Numeric input Items

To create numeric input items for the STB NIC 2212 example, beginning with a channel 1 input data word for the STB AVI 1270 module:

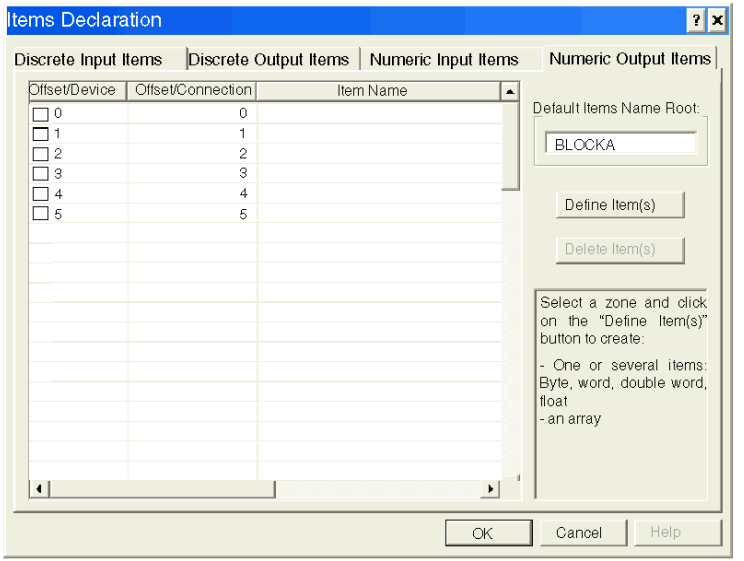
Step	Action
1	The Numeric Input Items tab looks like this:  **Note:** In this example, both the **Offset/Device** and **Offset/Connection** columns represent the byte address. All items you create will be either an 8-bit byte or a 16-bit word.
2	In the **Default Items Name Root** input box type: **AVI1270_CH1_inputdata**.

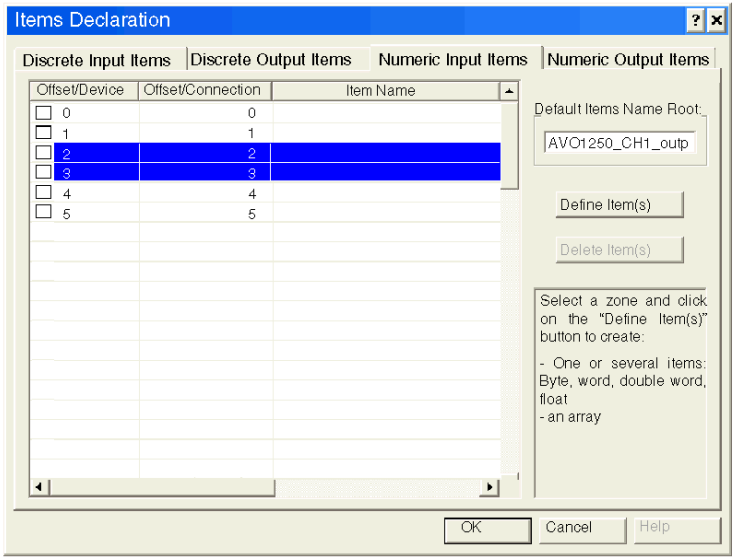
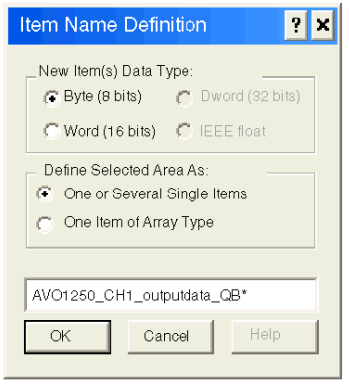
Step	Action
3	In the **Items List**, select bytes (or rows) 10 and 11: 
4	Click the **Define Item(s)** button.The **Item Name Definition** dialog opens: 

Step	Action
5	Select **Word (16 bits)** as the **New Item(s) Data Type**, then click **OK**. A new item is created: 
6	Repeat steps 2 - 5 for each group of numeric input item you need to create. In this example, that inlcudes items for each of the following groups: • Byte: 12, Default Items Name Root: AVI1270_CH1_inputstatus • Word: 14-15, Default Items Name Root: AVI1270_CH2_inputdata • Byte: 16, Default Items Name Root: AVI1270_CH2_inputstatus • Byte: 17, Default Items Name Root: AVO1250_CH1_outputstatus • Byte: 18, Default Items Name Root: AVO1250_CH2_outputstatus
7	Click on the **Numeric Output Items** tab to open that page.

Creating Numeric Output Items

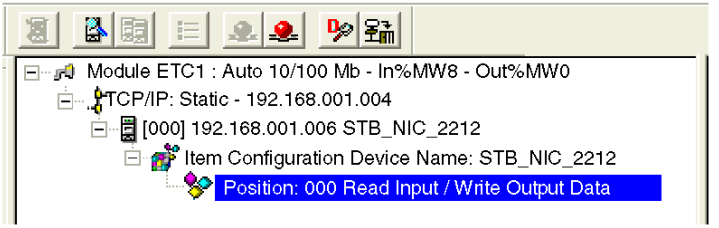
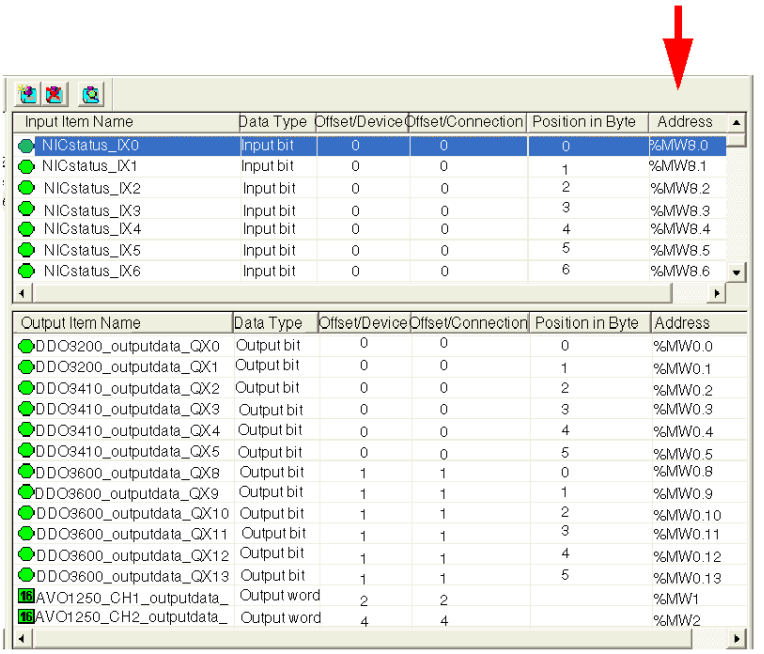
To create numeric output items for the NIC 2212, example, beginning with a output data word for the AVO 1250 module:

Step	Action
1	The Numeric Output Items tab opens:  **Note:** In this example, both the **Offset/Device** and **Offset/Connection** columns represent the byte address. All items you create will be 16-bit words comprising 2 bytes.
2	In the **Default Items Name Root** input box type: **AVO1250_CH1_outputdata**.

Step	Action
3	In the **Items List**, select bytes (or rows) 10 and 11: 
4	Click the **Define Item(s)** button.The **Item Name Definition** dialog opens: 
5	Repeat steps 2 - 5 for the AVO 1250 channel 2 output data at bytes 4 and 5.
6	Click **OK** to close the **Items Declaration** window.
7	Select **File** →**Save** to save your edits.

Viewing Input and Output Item Addresses

The final step in this example is to view the address the Unity Pro EtherNet/IP configuration tool assigns to each newly created input and output item. To do this:

Step	Action
1	In the **Devices** window of the Unity Pro EtherNet/IP configuration tool, the item at Position 000 should still be selected. If not, select it, as shown below:  The automatically configured input and output items appear on the right side of the screen in the I/O area (as depicted in the next step, below).
2	If necessary, use the horizontal scroll bar to scroll to the far right of the input or output area and display the *Address* column, which identifies the location of the input or output in the TSX ETC 100: 

Input Item Name	Data Type	Offset/Device	Offset/Connection	Position in Byte	Address
NICstatus_IX0	Input bit	0	0	0	%MW8.0
NICstatus_IX1	Input bit	0	0	1	%MW8.1
NICstatus_IX2	Input bit	0	0	2	%MW8.2
NICstatus_IX3	Input bit	0	0	3	%MW8.3
NICstatus_IX4	Input bit	0	0	4	%MW8.4
NICstatus_IX5	Input bit	0	0	5	%MW8.5
NICstatus_IX6	Input bit	0	0	6	%MW8.6

Output Item Name	Data Type	Offset/Device	Offset/Connection	Position in Byte	Address
DDO3200_outputdata_QX0	Output bit	0	0	0	%MW0.0
DDO3200_outputdata_QX1	Output bit	0	0	1	%MW0.1
DDO3410_outputdata_QX2	Output bit	0	0	2	%MW0.2
DDO3410_outputdata_QX3	Output bit	0	0	3	%MW0.3
DDO3410_outputdata_QX4	Output bit	0	0	4	%MW0.4
DDO3410_outputdata_QX5	Output bit	0	0	5	%MW0.5
DDO3600_outputdata_QX8	Output bit	1	1	0	%MW0.8
DDO3600_outputdata_QX9	Output bit	1	1	1	%MW0.9
DDO3600_outputdata_QX10	Output bit	1	1	2	%MW0.10
DDO3600_outputdata_QX11	Output bit	1	1	3	%MW0.11
DDO3600_outputdata_QX12	Output bit	1	1	4	%MW0.12
DDO3600_outputdata_QX13	Output bit	1	1	5	%MW0.13
AVO1250_CH1_outputdata_	Output word	2	2		%MW1
AVO1250_CH2_outputdata_	Output word	4	4		%MW2

3.4 Connecting to Third Party Devices

Overview

The EtherNet/IP communication module can connect to and communicate with EtherNet/IP devices made by third party manufacturers. This section describes how to set up communications with the Rockwell Automation 1734-AENT remote device and its I/O.

What Is in This Section?

This section contains the following topics:

Adding a Third Party Device to the Sample Network

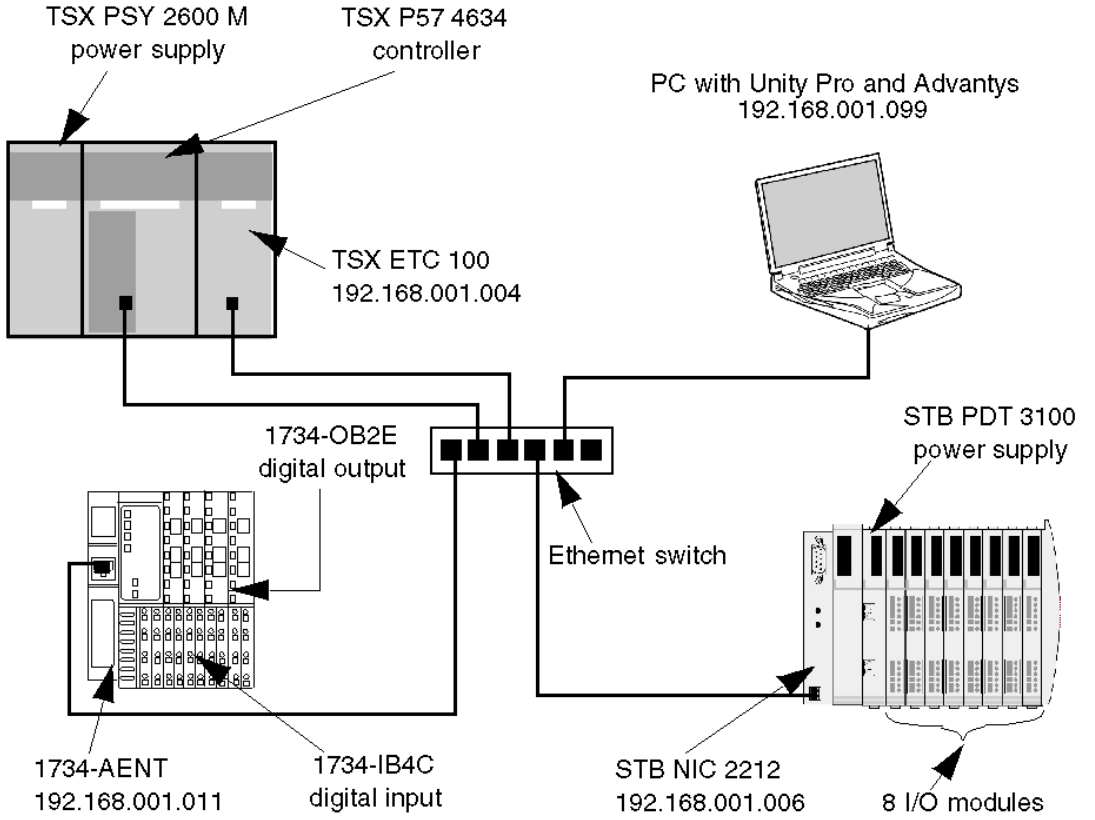
Overview

The next task is to extend the sample network by adding the following third party devices:

- 1734-AENT PointIO adapter with IP address of 192.168.001.011
- 1734-IB4/C 4pt DC input module
- 1734-OB2E 2pt DC output module

Network Topology

The modified network topology looks like this:



To re-create this example, be sure to:

- use the IP addresses for your own configuration's:
  - PC
  - TSX ETC 100 EtherNet/IP communication module
  - STB NIC 2212 EtherNet/IP network interface module
  - 1734-AENT PointIO adapter
- check all wiring

NOTE: Unity Pro software running in the PC is used to configure the TSX P57 4634 controller. In this example, the PC is indirectly wired to the CPU's Ethernet port via the Ethernet switch. Alternatively, you could bypass the switch and directly wire the PC to another one of the CPU's ports.

Add an EDS File

Overview

Before you can add a third party device to your configuration, be sure the EDS file for that device is included in the Unity Pro EtherNet/IP configuration tool's **Device Library**.

Use the EDS management wizard to add one or more EDS files to the **Device Library**. The wizard presents a series of instruction screens that:

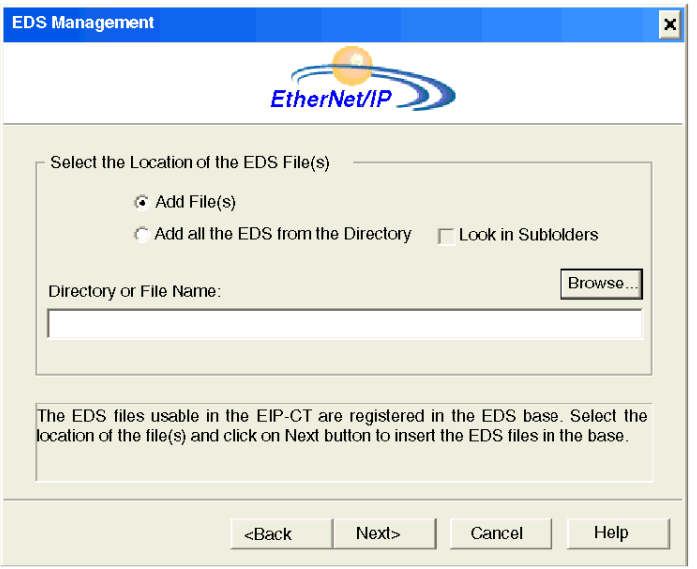
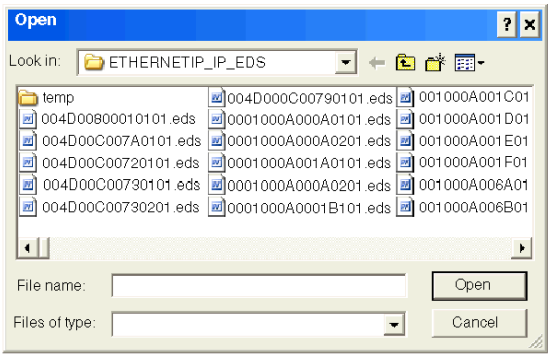
- simplify the process of adding EDS files to the **Device Library,** and
- provide a redundancy check that insures the same version of an EDS file cannot be added more than once

Note: Click **Devices →Options...** to open the **Display Options** window, where you can turn on or off the display of messages indicating the EDS file you are adding is a duplicate, or a different version of an EDS file already included in the **Device Library**.

Adding EDS Files

To add one or more EDS files to the **Device Library**:

Step	Action
1	Do one of the following: • in the **Device Library**, click the Add button  , or • in the **Library** menu, click **Add** Page 1 of the wizard opens.

Step	Action
2	Click **Next**. Page 2 of the wizard opens: 
3	In the **Select the Location of the EDS File(s)** section, select either: ● **Add File(s)**, to add one or more EDS files you will individually select, or ● **Add all the EDS Files from the Directory**, to add all files from a folder you will select. ● Select **Look in Subfolders** to also add EDS files in subfolders beneath the folder you select
4	Click the **Browse** button. The **Open** dialog opens: 

Step	Action
5	Use the **Open** dialog to navigate to and select: ● one or more EDS files, or ● a folder containing EDS files
6	Click **Open**. The dialog closes and your selection appears in the **Directory or File Name** field.
7	Click **Next**. The wizard compares the selected EDS files against existing files in the **Device Library**.
8	(Conditional) If one or more selected EDS files are duplicates and if notice of redundant files is enabled in the **Display Options** dialog, the configuration tool displays a **File Already Exists** message. **Close** the message.
9	Page 3 of the wizard opens indicating the Status of each device you selected ● a green check mark indicates the EDS file can be added ● a blue informational icon indicates a redundant file (Optional) Select a file in the list, then click **View Selected File** to open it.
10	Click **Next** to add the non-duplicate files. Page 4 of the wizard opens, indicating the action is complete.
11	Click **Finish** to close the wizard. The device(s) you added can now be inserted into your EtherNet/IP configuration.

Automatically Detect and Add the 1734-AENT PointIO Adapter

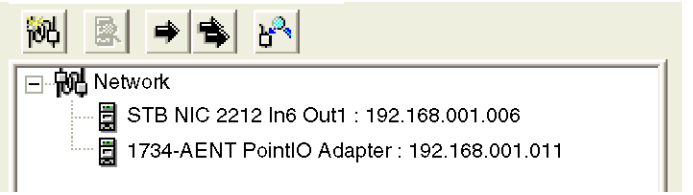
Overview

Use the Unity Pro EtherNet/IP configuration tool to automatically detect the 1734-AENT PointIO adapter. After it is detected, you can add it to your project.

NOTE: The 1734-AENT must be active online with a valid IP address before you can detect and add it to your project.

Detecting and Adding Network Devices

To automatically detect the 1734-AENT, then add it to your project, follow these steps:

Step	Action
1	Launch the configuration tool from the **Configuration** page of the EtherNet/IP communication module's **Properties** window.
2	In the configuration tool, begin on-line operations by clicking the **Go Online** button .
3	Click on the **Configuration** tab to enable automatic network detection: 
4	Click the **Read Network Configuration** toolbar button . The configuration tool searches the network for EtherNet/IP devices, classifies them using the device EDS file, then lists the EtherNet/IP devices it detects. 
5	Select the **1734-AENT PointIO Adapter** in **Network Detection** window.
6	Click the Insert in Configuration button . The **Properties** window opens, where you can configure the 1734-AENT PointIO adapter.

Configuring 1734-AENT PointIO Adapter Properties

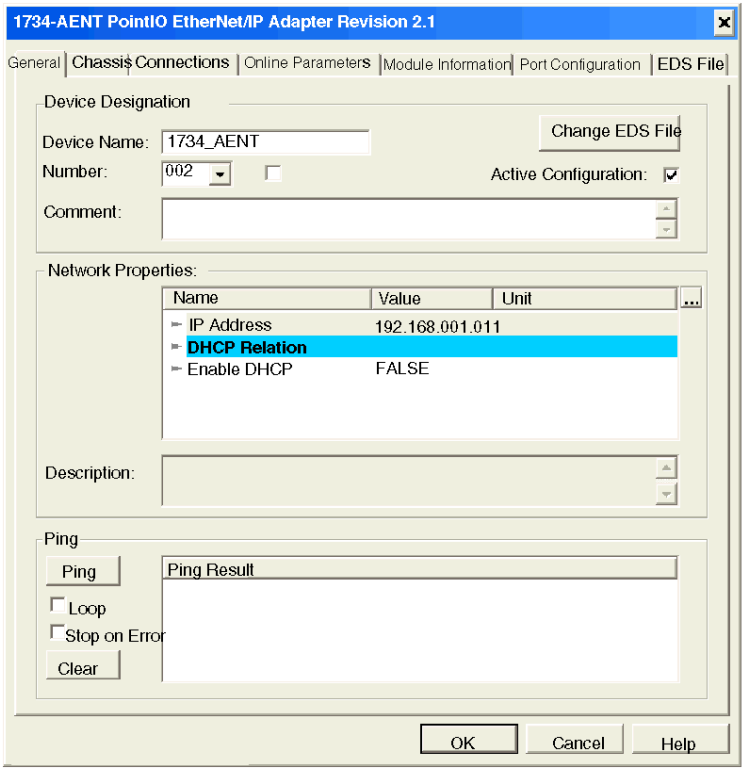
Overview

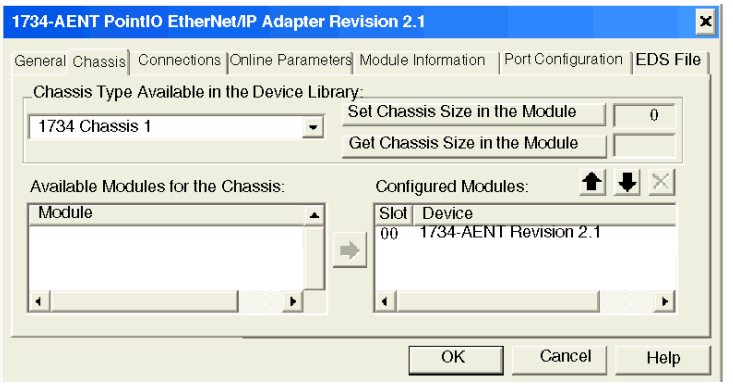
The 1734-AENT PointIO adapter module properties window presents the following tabbed pages. Only some of these pages need to be edited for this example:

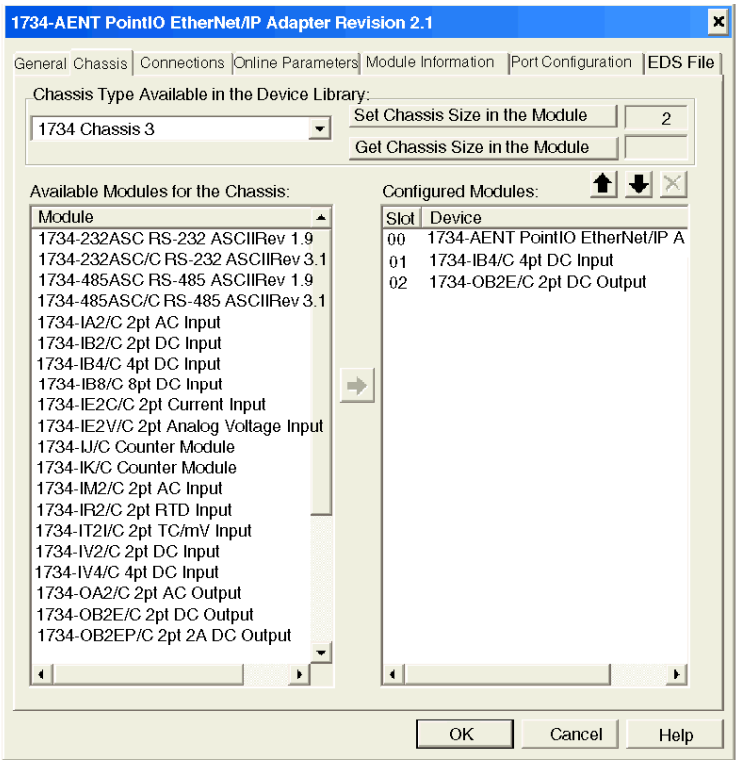
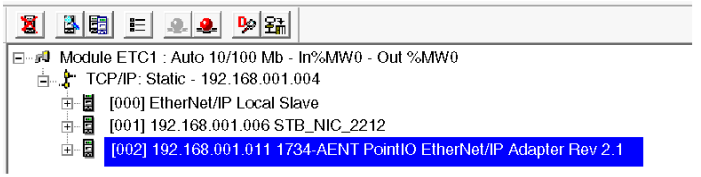
In this page...	Do the following...
General	• input device name • configure IP address • add the device to the project configuration
Chassis	Add 2 I/O modules to the chassis: • 1734-IB4/C 4pt DC input module • 1734-OB2E 2pt DC output module
Connections	Accept the default settings.
Online Parameters	Accept the default settings, if any.
Module Informations	(Read-only page - no configuration required)
Port Configuration	(Read-only page - no configuration required)
EDS File	(Read-only page - no configuration required)

Configuring the 1734-AENT

The following settings were used in the sample configuration:

<table>
<tr><th>Step</th><th colspan="2">Action</th></tr>
<tr><td>1</td><td colspan="2">Click on the General page:
 
</td></tr>
<tr><td rowspan="4">2</td><td colspan="2">In the General page, edit the following settings:</td></tr>
<tr><td>Device Name</td><td>1734_AENT</td></tr>
<tr><td>Number</td><td>The sequence of the device in the Devices window. for this example, type in 003.</td></tr>
<tr><td>Active Configuration</td><td>Be sure this checkbox is selected.</td></tr>
<tr><td></td><td>IP Address</td><td>192.168.001.011</td></tr>
</table>

Step	Action	
3	Click on the **Chassis** page: 	
4	In the **Chassis** page, do the following:	
	a	Select **1734 Chassis 3** in the **Chassis Type Available in the Device Library** list. The **Available Modules for the Chassis** list is populated and two [Empty] rows appear in the **Configured Modules** list.
	b	Select **1734-IB4/C 4pt DC Input** in the **Available Modules for the Chassis** list.
	c	Click the Insert button ➡. The module appears in position 01 in the **Configured Modules** list.
	d	Select **1734-OB2E/C 2pt DC Output** in the **Available Modules for the Chassis** list.
	e	Click the Insert button ➡. The module appears in position 02 in the **Configured Modules** list.

Step	Action
5	The configured **Chassis** page looks like this: 
6	Click **OK** to save your settings and close the properties window. A node is added to the project configuration in the **Devices** window:  The next step is to view the remote device's inputs and outputs.

Viewing 1734-AENT PointIO Adapter I/O Addresses

Overview

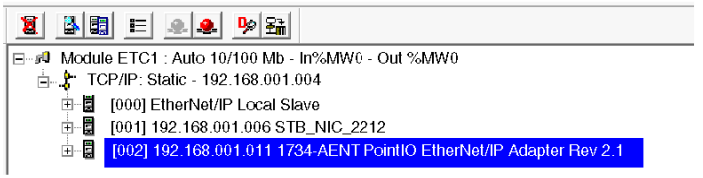
Because the Device Library includes EDS files for the 1734-AENT PointIO adapter and its discrete input and output modules, the Unity Pro EtherNet/IP configuration tool automatically:

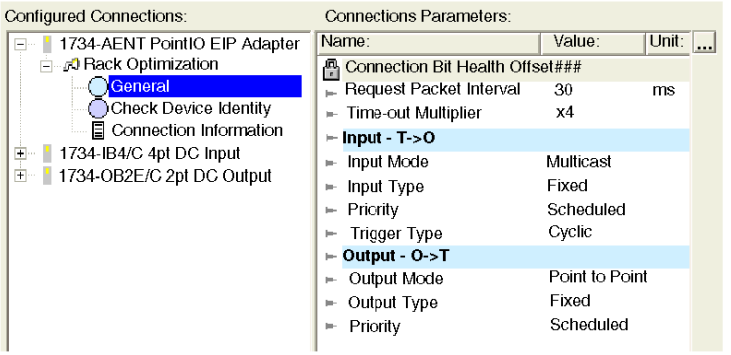
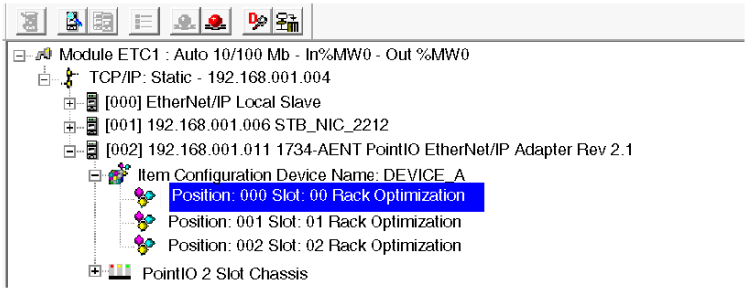
- creates a rack optimized CIP connection from the TSX ETC 100 EtherNet/IP communication module to the 1734-AENT, and
- configures each input and output item by assigning:
  - an item name
  - an address location
  - a size allotment based on its data type

NOTE: In this example, the configuration tool created a rack optimized connection, which is more efficient. A rack optimized connection can be used only with discrete (digital) I/O modules. For analog I/O modules, each analog module must be connected to the ETC 100 using a separate connection.

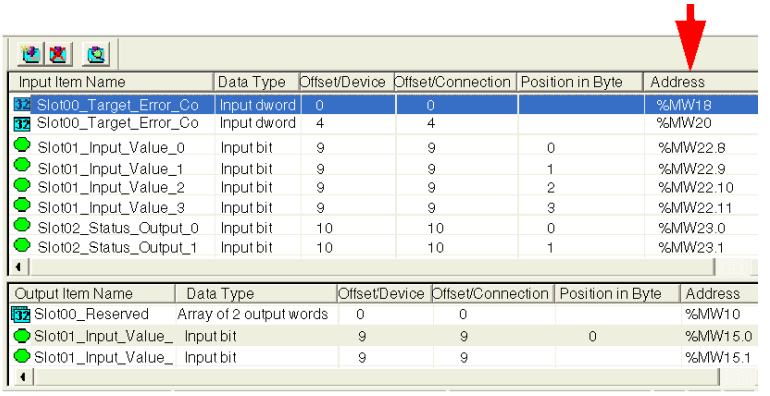
Viewing the CIP Connection and I/O

To view the automatically created CIP connection and the I/O items in the Unity Pro EtherNet/IP configuration tool:

Step	Action
1	In the **Devices** window, select the 1734-AENT: 
2	In the **Protocol** menu, select **Properties**. The **Properties** window opens for the remote device.

Step	Action
3	In the **Connections** tab, under the topmost Rack Optimization connection, select **General**. The rack optimized connection properties are displayed in the Connection Parameters list: 
4	In the **Devices** window, navigate to and select the first Rack Optimized connection item at Position 000, as shown below:  The automatically configured input and output items appear on the right side of the screen in the I/O area (as depicted in the next step, below).

Step	Action
5	If necessary, use the horizontal scroll bar to scroll to the far right of the input or output area and display the *Address* column, which identifies the location of the input or output in the TSX ETC 100:



Input Item Name	Data Type	Offset/Device	Offset/Connection	Position in Byte	Address
Slot00_Target_Error_Co	Input dword	0	0		%MW18
Slot00_Target_Error_Co	Input dword	4	4		%MW20
Slot01_Input_Value_0	Input bit	9	9	0	%MW22.8
Slot01_Input_Value_1	Input bit	9	9	1	%MW22.9
Slot01_Input_Value_2	Input bit	9	9	2	%MW22.10
Slot01_Input_Value_3	Input bit	9	9	3	%MW22.11
Slot02_Status_Output_0	Input bit	10	10	0	%MW23.0
Slot02_Status_Output_1	Input bit	10	10	1	%MW23.1

Output Item Name	Data Type	Offset/Device	Offset/Connection	Position in Byte	Address
Slot00_Reserved	Array of 2 output words	0	0		%MW10
Slot01_Input_Value_	Input bit	9	9	0	%MW15.0
Slot01_Input_Value_	Input bit	9	9		%MW15.1

Optimizing Performance

4

Overview

This chapter describes how to optimize performance of your EtherNet/IP network.

What Is in This Chapter?

This chapter contains the following sections:

4.1 Selecting a Switch

Overview

This section describes how to select an Ethernet switch for your network.

What Is in This Section?

This section contains the following topics:

Role of a Switch in an Ethernet Network

Overview

Schneider Electric recommends the use of managed switches—not unmanaged switches or hubs—in process control networks. A managed switch provides more functionality than an unmanaged switch, including the ability to:

- turn switch ports on or off
- configure port speed and duplex settings
- control and monitor message traffic within segments
- prioritize message traffic

Recommended Switch Features

When acquiring an Ethernet switch for your process control network, confirm that the switch includes the following features:

- Multiple speed (10/100/1000 Mbps)
- Full duplex
- QoS
- IGMP snooping
- RSTP
- VLAN support
- Port mirroring
- SNMP agent

Transmission Speed, Duplex and Auto-Negotiation

Introduction

Most Ethernet switches support multiple transmission speeds, full- and half-duplex communication, and offer auto-negotiation capability. Hubs, by contrast, are not designed to support full duplex transmissions.

Duplex

Full duplex enables a switch port to both transmit and receive messages simultaneously, over two dedicated communication channels. Half duplex, by contrast, permits a port to transmit or receive messages in only one direction at a time. Signal collisions are possible in half duplex communications—because messages are transmitted and received over a single channel. Half duplex communications can cause poor performance and message loss.

Auto-Negotiation

Auto-negotiation permits a switch port—connected to a remote device that also supports auto-negotiation—to automatically configure itself for the maximum speed and duplex configuration supported by both devices. However, it may be necessary to manually configure the speed and duplex settings of the switch port, if its peer device does not possess auto-negotiation capability.

Recommendation

Schneider Electric recommends that you employ only switches that support:

- both auto-negotiation and manual configuration of speed and duplex settings
- multiple speeds: 10/100/1000 Mbps
- both full duplex and half duplex

IGMP Snooping

Multicast Messaging

Internet Group Management Protocol (IGMP) is an essential feature of multicast messaging. IGMP instructs routers and switches to forward Ethernet multicast packets to only those device ports that have requested these packets.

In the absence of IGMP snooping, a switch forwards multicast packets out of all its ports, resulting in greater network traffic, wasted network bandwidth, and degraded network performance.

Configure one Ethernet network switch as the IGMP querier. This switch periodically polls the field devices connected to the network, which causes all connected devices to issue an *IGMP Multicast Group Join* message. The group message is received by all network switches, which update their multicast addressing information databases in response.

Similarly, when an Ethernet device transmits an *IGMP Multicast Group Leave* message, all network switches update their multicast addressing information databases by removing the device from their databases.

Multicast messaging reduces network traffic by:

- requiring that a message be sent only once
- sending the message only to devices for which the message is intended

Recommendation

Schneider Electric recommend the following:

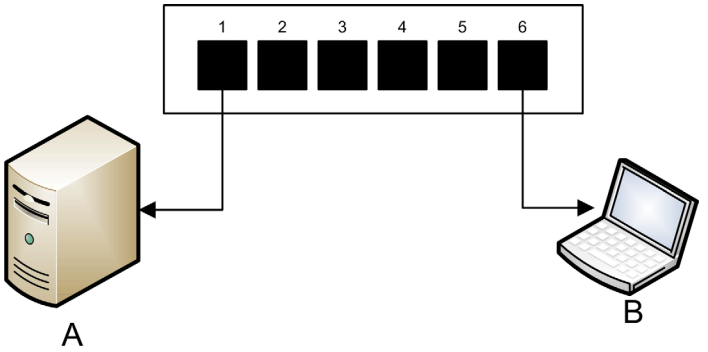
- employ switches that support IGMP V2 or higher
- because IGMP snooping may be disabled by default, enable IGMP snooping for each network switch
- confirm that one switch is configured as the IGMP querier

Port Mirroring

Introduction

Port mirroring lets you troubleshoot switch port transmissions by copying the traffic that passes through one port (the source or mirrored port) and sending the copied transmission to a second port (the destination or mirror) port, where the packets can be examined.

In the following example, the data packets transmitted over port 1 are copied and sent to port 6. To troubleshoot port 1, a PC with packet sniffing software is used to analyze the traffic on port 6 and thereby troubleshoot port 1.



A target device of port 1 transmissions
B PC with packet sniffing software connected to port 6, which mirrors port 1 transmissions

Port mirroring does not affect the normal forwarding behavior of the mirrored port. In many switches, you can configure port mirroring so that you can forward and examine:

- only the incoming packets of a single mirrored port
- only the outgoing packets of a single mirrored port
- both the incoming and outgoing packets of a single mirrored port
- the packets of several mirrored ports—or the whole switch

A packet sniffer's troubleshooting features should include:

- analyzing network performance
- monitoring network activity

Recommendation

Schneider Electric recommends implementing port mirroring as follows:

- Use a destination or mirror port only for port mirroring and not for any other purpose. Connect only the PC with packet sniffer to the mirroring port.
- When configuring the switch, confirm that port mirroring is designed to forward packets—e.g., incoming, outgoing, or both—to meet your requirements.
- A packet sniffer's troubleshooting features should include the capabilities of analyzing network performance and monitoring network activity.

Virtual Local Area Network (VLAN)

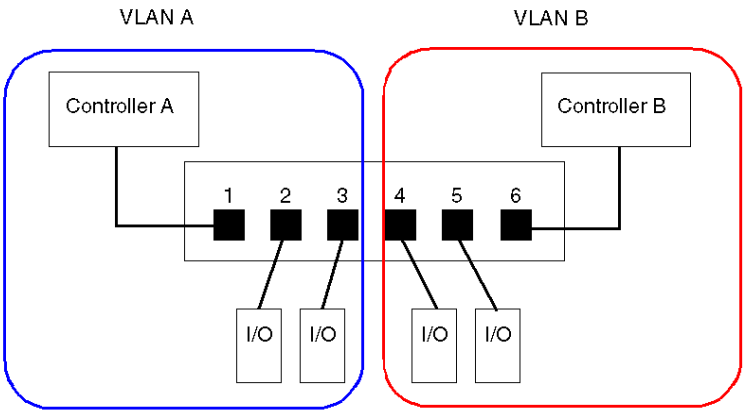
A VLAN is a group of Ethernet devices—which may be physically located on different network segments—that are grouped together and communicate as if they were located on the same LAN segment. All devices on a VLAN use the same IP subnet

In a port-based—or static—VLAN, an administrator configures VLAN membership by assigning individual switch ports to the VLAN. Any device connected to that port is effectively added to the VLAN.

NOTE: A single port can be a member of multiple VLANs.

VLANs permit the creation of logically separate groups of network devices, without having to physically re-wire those devices. When a switch receives a message directed to a specific VLAN, it forwards that message only to the switch ports that are members of that VLAN. The switch does not send the message to its ports that are not members of that VLAN.

In the example, below, switch ports 1, 2, and 3 are assigned to VLAN A, while switch ports 4, 5, and 6 are assigned to VLAN B:



Simple Network Management Protocol (SNMP) Agent

An *SNMP agent* is a software component that responds to queries about the management data of the switch, and reports events to another device acting as an SNMP manager.

The management data for a switch can include:

- operational state information (interface status, mode of operation, etc.)
- configuration parameters (IP address, features enabled / disabled, timer values, etc.)
- performance statistics (frame counters, event logs, etc.)

If a switch is equipped with SNMP agent software, a designated SNMP manager can:

- retrieve management data about the switch
- control the switch by editing its configuration settings
- receive traps—or notices of events—affecting the state of the switch

4.2 Control Application Design

Overview

In a control system, control and automation are achieved by processing and delivering various application service messages.

Understanding messages, allocating network bandwidth among messages, and determining the time required for a message to traverse the network are all major performance considerations of your control application design.

What Is in This Section?

This section contains the following topics:

Message Types

Overview

Two types of industrial Ethernet message types are supported by the Ethernet communication module:

Message Type	Includes...
Explicit	• Non-time critical management data • Read/write application data
Implicit	• Real-time I/O data • Real-time control data • Real-time synchronization data

Explicit Messages

Explicit messages transmit information used for device configuration and diagnostics, and for data collection. In explicit messaging, the client issues a request; the server receives, processes, and sends a response back to the client.

You can specify a response timeout value, indicating how long the client waits for a response from the server. If the client does not receive a response from the server within the response timeout period, the client reissues its request. The length of the response timeout will vary depending on the requirements of your application.

Examples of explicit messages include: SNMP messages, FTP messages, CIP establish connection messages, EtherNet/IP query and response messages, and DHCP messages.

The characteristics of explicit messaging are:
- point-to-point client-server mode
- variable size
- variable frequency
- long response time
- long connection timeout

Explicit messages can be sent as either connected or unconnected, depending on the frequency of your need for data, and on the level of service required:

Message type	Characteristics
Connected	• Begins when an originating device initiates a connection by sending a request to a target device. • The connection is established when the originator receives a successful response from the target. • A CIP connected message has a higher priority and provides better service, but requires a greater amount of resources from both the target and originator devices. • Used for recurring requests, and for high priority parameter monitoring. • Typically use short response timeout settings.
Unconnected	• Less resource intensive. • Used for less frequent requests, and for lower priority parameter monitoring. • Typically use very long response timeout settings.

NOTE: The response timeout can be configured using the **EM Request Timeout** parameter (located in the **Channel Properties** →**EtherNet/IP** page).

Implicit Messages

Implicit messages consist of packets of data that are time critical. Implicit messages are used for real-time control and synchronization. Examples of implicit messages include: real-time I/O data, motion control data, functional diagnostic data, real-time synchronization data, and network topology management data.

Implicit messages require determinism and high performance in message processing and delivery.

The characteristics of implicit messaging are:
- producer/consumer mode (EtherNet/IP) or client/server mode (Modbus TCP)
- small, fixed data size
- fixed frequency
- short response time
- short connection timeout

TCP Connections

Overview

EtherNet/IP uses TCP connections as a pipeline for CIP connections Both connected and unconnected messaging use the TCP connection.

TCP Connection Limits

The TSX ETC 100 EtherNet/IP communication module can provide up to 67 TCP connections, as follows:

Connection type	Maximum number of connections
I/O adapter	64[1]
I/O scanner	
Explicit message client	
Explicit message server	3
Total TCP connections:	67

[1] 64 connections can be used for any combination of:
- I/O adapter connections
- I/O scanner connections
- explicit messages (as client)

A single TCP connection can support multiple CIP connections.

NOTE: TCP connections dedicated to other services, for example FTP, are not included in the above numbers.

CIP Connections and Messages

Overview

EtherNet/IP uses CIP connections to transmit messages between objects running in connected devices. There are different types of CIP connections.

Connection Types

CIP connection types include:

CIP connection type...	Supports...
Rack optimized	The grouping of data from multiple, I/O modules in the same rack transmitted over a single connection. **Note:** A rack optimized connection: • can transmit only device status and data • applies only to digital I/O modules A CIP connection is consumed by each I/O module, in addition to the rack optimized connection.
Direct	A link between a controller and a single device. **Note:** A connection to an analog I/O module must be via a direct connection.

Connection Limits

The TSX ETC 100 EtherNet/IP communication module can provide up to 198 concurrent CIP connections, as follows:

Connection type	Maximum number of connections
I/O adapter	128[1]
I/O scanner	
Explicit message client	64
Explicit message server	6
Total TCP connections:	198
[1]128 CIP connections can be used for any combination of: • I/O adapter connections • I/O scanner connections	

NOTE: Up to16 simultaneous explicit messaging connections can be active per scan.

Messaging Performance

Maximum Messaging Load

The TSX ETC 100 EtherNet/IP communication module supports an implicit messaging load of up to 7500 data packets per second (pps).

4.3 Projecting Ethernet Network Performance

Overview

This section presents an example of how to calculate the impact of your project on Ethernet network performance.

What Is in This Section?

This section contains the following topics:

Allocating Network Bandwidth

Introduction

Maximum network bandwidth equals your network speed, for example 100 Mbps. When designing your control network, allocate network bandwidth among the control application messages required by your application.

NOTE: Schneider Electric recommends you reserve at least the following amounts for processing explicit messaging:

- 10% of network bandwidth
- 10% of CPU processing capacity for each network device

Message Load and Message Bandwidth

Message Load—in packets per second (PPS)—represents the number of packets in a single message that are received and sent within one second. *Message Load* can be estimated as follows:

Message Load =

(number of packets per connection) x (number of connections) / RPI

The *number of packets per connection* value depends on the capacity of the device, and can be either:

- 1: for connections that support uni-directional communication
- 2: for connections that support input and output (for producer/consumer mode) or request and response (for client/server mode) per one time bi-directional exchange, or

The connection can be used for either explicit or implicit messaging. For UDP-based explicit messaging, assume that each client represents one connection, and that messages are transmitted cyclically.

Message Bandwidth (in bits) can be calculated as follows:

Message Bandwidth = message packet size (bits) x Message Load

Based on the portion of network bandwidth you want to allocate to a particular message, you can use the *Message Load* and *Message Bandwidth* formulae to calculate the fastest RPI for the message.

Device Load and Device Bandwidth

Device Load—measured in number of packets—represents the load contributed by messages received and sent by a device within one second. *Device Load* is the sum of the *Message Load* values for every message handled by the device.

If the *Device Load* exceeds the device's processing capability, performance of both the device and the network is degraded.

NOTE: Schneider Electric recommends that *Device Load* not exceed 90% of CPU processing capacity of each device.

Device Bandwidth—measured in bits—is the sum of the *Message Bandwidth* values for messages handled by the device

In your control application design, determine whether the I/O scanner device can handle the load contributed by every I/O adapter device. To do this, perform the following steps:

1. Calculate the implicit messaging load and bandwidth for each remote device.
2. Sum the load and bandwidth estimates for every remote device.
3. Compare the total implicit messaging load and bandwidth against the maximum implicit messaging capacity of the device acting as I/O scanner.

If the projected total load or bandwidth for a communication module acting as an I/O scanner exceeds its implicit messaging load or bandwidth limits, consider one or more of the following corrective actions:

- If the I/O adapter supports rack optimized connections, and if a single rack of digital I/O uses multiple direct connections, replace the direct connections with a single rack optimized connection, if possible.
- Increase the RPI setting for a device where possible.
- Add another communication module to act as an I/O scanner, and re-design the network in order to share the load.

Network Load and Network Bandwidth

Network Load—measured in number of packets—can be estimated as the sum of the *Device Load* of the adapter devices, or of the scanner devices.

Network Bandwidth—measured in bits—can be estimated as the sum of the *Device Bandwidth* of the adapter devices, or of the scanner devices.

NOTE: Schneider Electric recommends that *Network Load* not exceed 90% of maximum network bandwidth.

If necessary, you may need to optimize your control application design by:

- adjusting device RPI settings
- changing connection types (e.g., from direct to rack optimized)
- modify the configuration
- change the network topology

Network Load and Bandwidth Calculation Example

Network Devices

This example estimates the performance for an Ethernet network composed of the following devices:

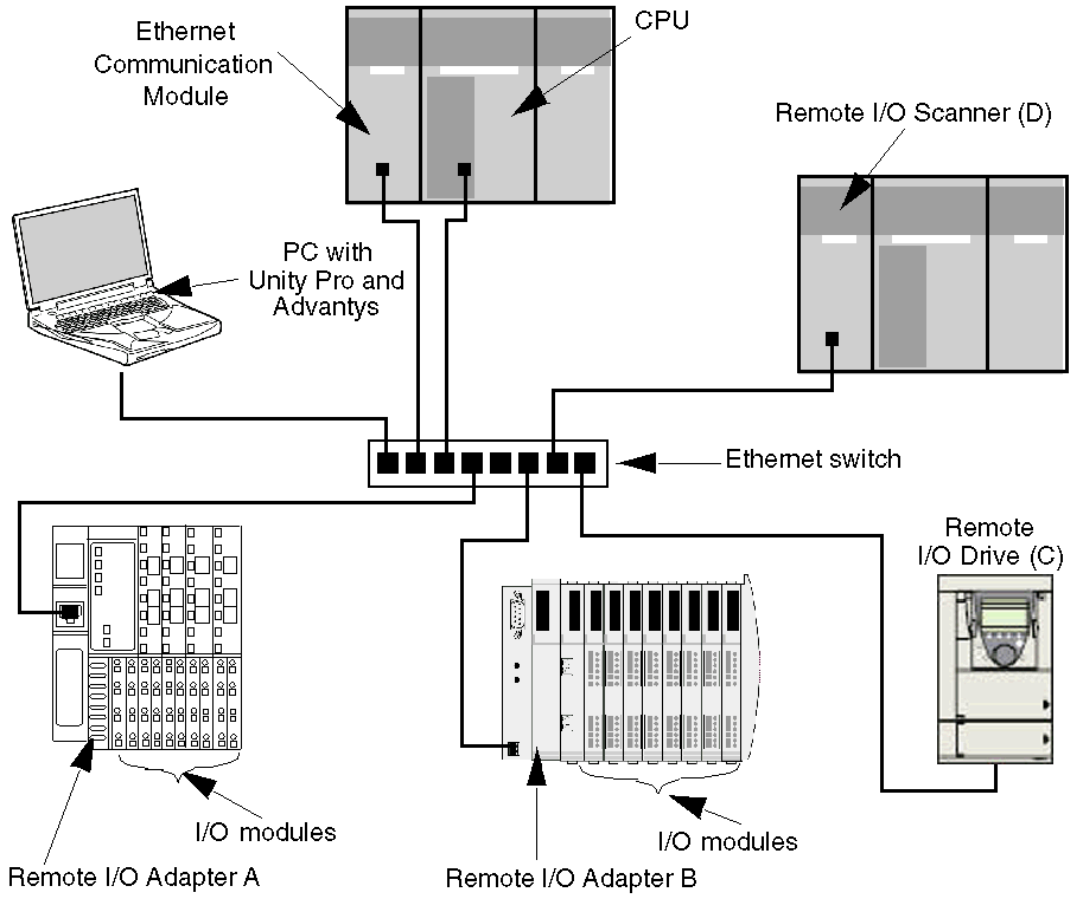
- a PLC that controls 3 remote I/O stations (A, B, and C)
- Ethernet communication module, acting as the local I/O scanner, installed in the PLC rack
- an 8-port Ethernet managed switch
- a PC running used to obtain diagnostic data via explicit messages running the following software:
  - Unity Pro
  - the Unity Pro Ethernet Configuration Tool
- 4 remote devices, acting as:
  - an I/O adapter (A) for a rack of I/O modules
  - a second I/O adapter (B) for a rack of I/O modules
  - a remote I/O drive (C)
  - a remote I/O scanner (D)

Unity Pro software running in the PC is used to configure the CPU controller.

For programming purposes you need a connection to the PLC either through the CPU's Ethernet port or other supported programming paths.

Network Diagram

The proposed network diagram looks like this:



Network Load and Bandwidth Limits

When performing calculations, keep in mind that the Ethernet module and remote devices cannot exceed their implicit messaging and bandwidth limits:

Device	Load Limits	Bandwidth Limits
Ethernet Communication Module	12000 pps	80 Mbps
I/O Adapter (A)	8000 pps	70 Mbps
I/O Adapter (B)	8000 pps	70 Mbps

Device	Load Limits	Bandwidth Limits
I/O Drive (C)	8000 pps	70 Mbps
I/O Scanner (D)	12000 pps	80 Mbps
Switch	16000 pps	90 Mbps

Remote Device Connections and RPI

For the purpose of this example, it is assumed that the remote devices require the following numbers of CIP connections, and are configured for the stated requested packet interval (RPI) settings:

Device	CIP I/O Connections	RPI Setting	I/O Packet Size
I/O Adapter (A)	5	20 ms	8000 bits
I/O Adapter (B)	2	30 ms	4096 bits
I/O Drive (C)	2	30 ms	8000 bits
I/O Scanner (D)	2	50 ms	8000 bits

For the purposes of this example, it is also assumed that every connection is bi-directional.

I/O Scanner Calculations

The Ethernet communication module, acting as local I/O scanner, has to handle the implicit messaging load contributed by the remote devices. Your task is to:

1. estimate the implicit messaging load and bandwidth contributed by each remote device
2. sum the load and bandwidth values for each remote device
3. compare the total load and bandwidth against the maximum implicit messaging capacity of the local I/O scanner

Recall that the implicit messaging load calculation formula for a single remote device is:

Load = (number of packets per connection) x (number of connections) / RPI

Because every connection is assumed to be bi-directional, the *number of packets per connection* value is 2. Consequently, the estimated implicit messaging load contributed by each device, and the total implicit messaging load the local I/O scanner has to handle can be estimated as follows:

Load:

Device	Number of packets per connection	X	Number of connections	÷	RPI	=	Load
I/O Adapter (A)	2	X	5	÷	20 ms	=	500 pps
I/O Adapter (B)	2	X	2	÷	30 ms	=	134 pps
I/O Drive (C)	2	X	2	÷	30 ms	=	134 pps
I/O Scanner (D)	2	X	2	÷	50 ms	=	80 pps
Total						=	848 pps
Switch						=	848 pps

Bandwidth:

Device	Packet size	X	Load	=	Bandwidth
I/O Adapter (A)	8000 bits	X	500 pps	=	4 Mbps
I/O Adapter (B)	4096 bits	X	134 pps	=	0.554 Mbps
I/O Drive (C)	8000 bits	X	134 pps	=	1.07 Mbps
I/O Scanner (D)	8000 bits	X	80 pps	=	0.64 Mbps
Total				=	6.26 Mbps
Switch				=	6.26 Mbps

Conclusion

The projected total load for the module—848 pps—is within the device implicit messaging limit of 12000 data packets per second. The projected total bandwidth for the communication module—6.26 Mbps—is also within the device implicit messaging bandwidth limit of 80 Mbps. The projected total load and bandwidth for the remote devices (including the switch) are also within their 90% load and bandwidth limits:

Device	90% of Load Limit	90% of Bandwidth Limit
Ethernet Communication Module	10800 pps	72 Mbps
I/O Adapter (A)	7200 pps	63 Mbps
I/O Adapter (B)	7200 pps	63 Mbps
I/O Drive (C)	7200 pps	63 Mbps
I/O Scanner (D)	10800 pps	72 Mbps

NOTE: Although message load contributed by explicit messaging are not included in the above calculations, such load contributions are presumed to be less than 10% of the device load and bandwidth.

Explicit Messaging In Unity Pro

5

Overview

This chapter describes how to execute explicit messages in Unity Pro using either:

- the SEND_REQ function block
- the *Online Action* window of the Unity Pro EtherNet/IP configuration tool

Unity Pro supports both connected and unconnected explicit messaging.

NOTE: Your Unity Pro application can contain more than 16 explicit messaging blocks, but only 16 explicit messaging blocks can be active at the same time. Also, there can be only one concurrent explicit message - connected or unconnected - from an EtherNet/IP communication module to the same remote EtherNet/IP device.

What Is in This Chapter?

This chapter contains the following topics:

Explicit Messaging Services

Overview

Every explicit message performs a service. Each service is associated with a service code (or number). You will need to identify the explicit messaging service by its name, decimal number, or hexadecimal number.

You can execute explicit messages using either the SEND_REQ function block in Unity Pro, or the Unity Pro EtherNet/IP configuration tool.

Services

The services available in Unity Pro include, but are not limited to, the services listed below:

Service Code		Description	Available in...	
Hex	**Dec**		**Function Block**	**EIP config tool**
0	0	(*Reserved*)	—	—
1	1	Get_Attributes_All	X	X
2	2	Set_Attributes_All	X	X
3	3	Get_Attribute_List	X	—
4	4	Set_Attribute_List	X	—
5	5	Reset	X	X
6	6	Start	X	X
7	7	Stop	X	X
8	8	Create	X	X
9	9	Delete	X	X
A	10	Multiple_Service_Packet	X	—
B-C	11-12	(*Reserved*)	—	—
D	13	Apply_Attributes	X	X
E	14	Get_Attribute_Single	X	X
F	15	(*Reserved*)	—	—
10	16	Set_Attribute_Single	X	X
11	17	Find_Next_Object_Instance	X	X
12-13	18-19	(*Reserved*)	—	—
14	20	Error Response (DeviceNet only)	—	—
15	21	Restore	X	X
16	22	Save	X	X
"X" indicates the service is available. "—" indicates the service is not available.				

Service Code		Description	Available in...	
Hex	**Dec**		**Function Block**	**EIP config tool**
17	23	No Operation (NOP)	X	X
18	24	Get_Member	X	X
19	25	Set_Member	X	X
1A	26	Insert_Member	X	X
1B	27	Remove_Member	X	X
1C	28	GroupSync	X	—
1D-31	29-49	(*Reserved*)	—	—
"X" indicates the service is available. "—" indicates the service is not available.				

Configuring Explicit Messaging Using SEND_REQ

Overview

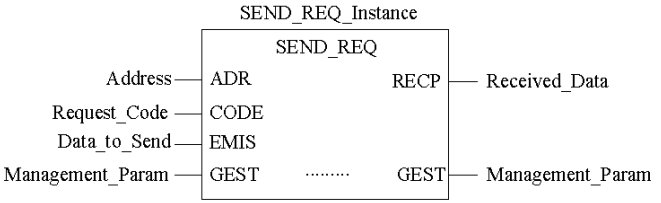
Use the SEND_REQ function block to configure EtherNet/IP connected and unconnected explicit messages. The SEND_REQ block can:

- send requests up to 252 bytes long, and
- receive responses up to 255 bytes long

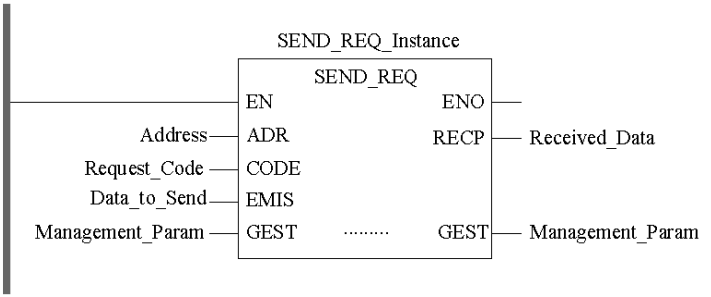
The Management_Param, Data_to_Send, and Received_Data parameters define the operation. Refer to Configuring the Management Parameter, *(see page 154)* Configuring the Data to Send Parameter *(see page 155)*, and Contents of the Received Data Parameter *(see page 155)*, below, for details.

EN and ENO can be configured as additional parameters.

FBD Representation



LD Representation



IL Representation

LD Address

SEND_REQ Request_Code, Data_to_Send, Management_Param, Received_Data

ST Representation

SEND_REQ (Address, Request_Code, Data_to_Send, Management_Param, Received_Data);

Input Parameters

Parameter	Data type	Description
Address	Array [0...5] of INT	The path to the destination device.
Request_Code	INT	Always 0x0E (for a CIP request).
Data_to_Send	Array [n...m] of INT	The message type (connected or unconnected) plus the CIP request.

Input/Output Parameters

Parameter	Data type	Description
Management_Param	Array [0...3] of INT	The management parameter, consisting of 4 words.

Output Parameters

Parameter	Data type	Description
Received_Data	Array [n...m] of INT	The CIP response.

Configuring the Address Parameter

To configure the Address parameter, use the ADDR function to convert a character string to an address, as follows:

ADDR('{network.station}rack.module.channel.destination address')

NOTE:

- The Xway address elements {network.station} are required only when bridging through another PLC station.
- The channel parameter value is always 0.

Configuring the Management Parameter

The Management Parameter consists of 4 contiguous words, as described below:

Data source	Register	Description	
		High Byte (MSB)	**Low Byte (LSB)**
Data managed by the system	`Management_Param[0]`	Exchange number	Activity Bit (bit 0) (see below).
	`Management_Param[1]`	Operation report *(see page 157)*	Communication report *(see page 156)*

Data managed by the user	`Management_Param[2]`	Block timeout. Values include:
		• 0 = infinite wait
		• other values = timeout x 100 ms, for example:
		  • 1 = 100 ms
		  • 2 = 200 ms
		  • ...etc.	

	`Management_Param[3]`	Length of data send/receive parameter: • Input (before send): length of `Data_to_Send` parameter • Output (after receive): length of `Received_Data` parameter	

Activity Bit:

This bit indicates the execution status of the communication function.

It is set to 1 when launched, and returns to 0 when its execution is complete.

It is the first bit of the first element of the table.

Example: if the management table has been declared as follows:

```
Tab_Gest ARRAY [0.3] OF INT,
```

the activity bit is the bit with the notation `Tab_Gest[1].0`.

NOTE: The notation previously used requires configuration of the project properties in such a way as to authorize the extraction of bits on integer types. If this is not the case, `Tab_Gest[1].0` cannot be accessed in this manner.

Configuring the Data_to_Send Parameter

The `Data_to_Send` parameter varies in size. It consists of contiguous registers that include—in sequence—both the message type and the CIP request:

Content	Length	Byte Offset	Data Type	Description
Message Type	1 word	0	INT	• 0 = unconnected message • 1 = connected message
CIP Request[1]	Set by `Management_Param[3]` (size of `Data_to_Send`)	2	Byte	**Service**: the service performed by the explicit message.
		3	Byte	**Request_Path_Size**: the number of words in the Request_Path field.
		4	Byte Array	**Request_Path**: The path of the request—including class ID, instance ID, etc.—for this transaction.
		...	Byte Array	**Request_Data**: Service specific data to be delivered in the explicit message request—if none, this field is empty
1 The CIP request must be structured in little endian order.				

Contents of the Received_Data Parameter

The `Received_Data` parameter contains only the CIP response. The length of the CIP response varies, and is reported by `Management_Param[3]` after the response is received. The format of the CIP response is described, below:

Byte offset	Field	Data type	Description
0	Reply Service	Byte	Service of the explicit message + 0x80
1	Reserved	Byte	0
2	General Status	Byte	EtherNet/IP General Status
3	Size of Additional Status	Byte	Additional Status array size—in words
4	Additional Status	Byte array	Additional status
...	Response Data	Byte array	Response data from request, or additional error data if General Status indicates an error

NOTE: The response must be structured in little endian order.

SEND_REQ: Communication and Operation Reports

Overview

Communication and operation reports are part of the management parameters.

NOTE: It is recommended that communication function reports always be tested at the end of their execution and before the next activation. On cold start-up, it is imperative that all communication function management parameters be checked and reset to 0.

It may be helpful to use the %S21 to examine the first cycle after a cold or warm start. For more information, refer to Unity Pro online help for %S21.

Communication Report

This report is common to all functions. It is significant when the value of the activity bit switches from 1 to 0. The reports with a value between 16#01 and 16#FE concern errors detected by the processor that executed the function.

The different values of this report are indicated in the following table:

Value	Communication report (least significant byte)
16#00	Correct exchange
16#01	Exchange stop on timeout
16#02	Exchange stop on user request (CANCEL)
16#03	Incorrect address format
16#04	Incorrect destination address
16#05	Incorrect management parameter format
16#06	Incorrect specific parameters
16#07	Problem in sending to the destination
16#08	Reserved
16#09	Insufficient receive buffer size
16#0A	Insufficient send buffer size
16#0B	No processor system resources
16#0C	Incorrect exchange number
16#0D	No telegram received
16#0E	Incorrect length
16#0F	Telegram service not configured
16#10	Network module missing
16#11	Request missing
16#12	Application server already active

Value	Communication report (least significant byte)
16#13	UNI-TE V2 transaction number incorrect
16#FF	Message refused

NOTE: The function can detect a parameter error before activating the exchange. In this case the activity bit remains at 0, and the report is initialized with values corresponding to the error.

Operation Report

This report byte is specific to each function, and specifies the result of the operation on the remote application.

It is significant only if the communication report has the following values:

- 16#00 (correct exchange),
- 16#FF (message refused).

If the value of the communication report is 16#00, the operation report will have the following values:

Value	Operation report (most significant byte)
16#00	Positive result
16#01	Request not processed
16#02	Incorrect response
16#03	Reserved
Other Values:	
Request code + 16#30	Upon positive reply for certain requests
16#FE	Upon positive reply for certain requests
16#FB	Upon reply to a minor request

If the value of the communication report is 16#FF, the operation report will have the following values:

Value	Operation report (most significant byte)
16#01	No resources towards the processor
16#02	No line resources
16#03	No device or device without resources **(*)**
16#04	Line error
16#05	Length error
16#06	Faulty communication channel
16#07	Addressing error
16#08	Application error

Value	Operation report (most significant byte)
16#0B	No system resources
16#0C	Communication function not active
16#0D	Destination missing. For a CIP request, it is detected on a request timeout.
16#0F	Intra-station routing problem or channel not configured
16#11	Address format not managed
16#12	No destination resources. For a CIP request, this can occur when the device number is not part of the configuration.
16#14	Non-operational connection (example: Ethernet TCP/IP)
16#15	No resource on the local channel. For a CIP request, it may be that there are no resources to handle the message; or it is an internal error: no buffer available, no link available, impossible to send message.
16#16	Access not authorized (example: Ethernet TCP/IP)
16#17	Inconsistent network configuration (example: Ethernet TCP/IP)
16#18	Connection temporarily unavailable: For a CIP request, it may be that another explicit message is in progress for this device, or a TCP connection or encapsulation session is in progress.
16#21	Application server stopped
16#30	Transmission error. For a connected CIP request, it may occur when the connection opening times out.
16#80...16#87: Forward_Open response errors:	
16#80	Internal error
16#81	Configuration error: the length of the explicit message, or the RPI rate, needs to be adjusted
16#82	Device error: target device does not support this service
16#83	Device resource issue: no resource is available to open the connection
16#84	System resource issue: unable to reach the device
16#85	Data sheet error: incorrect EDS file
16#86	Invalid connection size
16#90...16#9F: Register session response errors	
16#90	Target device does not have sufficient resources
16#98	Target device does not recognize message encapsulation header
16#9F	Unknown error from target
Legend:	
(*)	Code only managed by PCMCIA cards: TSX FPP20 and TSX FPP10

SEND_REQ Example - Get_Attributes_Single

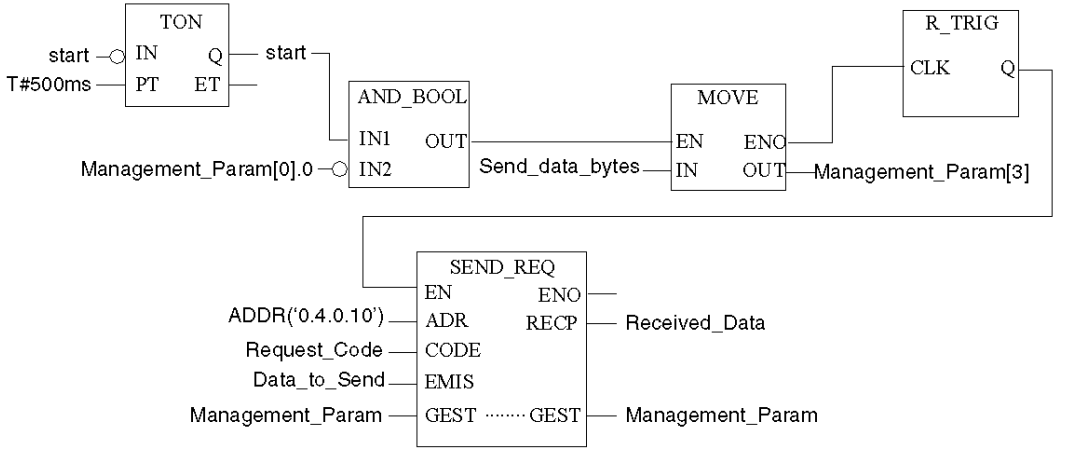
Overview

The following unconnected explicit messaging example shows you how to use the SEND_REQ function block to retrieve the vendor ID from a third-party device—in this case a Rockwell Automation 1734-AENT PointIO EtherNet/IP adapter (revision 2.1)—using the Get_Attributes_Single service.

You can perform the same explicit messaging service using the **Online Action** window of the Unity Pro EtherNet/IP configuration tool *(see page 169)*.

Implementing the SEND_REQ Function Block

To implement the SEND_REQ function block, you need to create and assign variables for the following blocks, as follows:



Input Variables

Variables need to be created and assigned to input pins. For the purpose of this example, variables have been created—and named—as described below. (You can, of course, use different variable names in your explicit messaging configurations.)

Input pin	Variable	Data type
IN	start	BOOL
IN	Send_data_bytes	INT
CODE	Request_Code	INT
EMIS	Data_to_Send	Array [0...4] of 5 INT

Note: The value of Send_data_bytes equals the length of the Data_to_Send variable, in bytes. In this example, Send_data_bytes = 16#000A.

Input/Output Variables

A single variable needs to be created and assigned to the dual input/output GEST pins. For the purpose of this example, a variable has been created—and named—as described below. (You can, of course, use different variable names in your explicit messaging configurations.)

Input pin	Variable	Data type
GEST	Management_Param	Array [0...3] of 4 INT

Output Variables

A variable also needs to be created and assigned to the single RECP output pin. (The names assigned to the output variable apply only to this example, and can be changed in your explicit messaging configurations.)

Output pin	Variable	Data type
RECP	Received_Data	Array [0...3] of 4 INT

Configuring the Address Variable

The Address variable identifies the explicit message originating device—-in this example, the TSX ETC 100 communication module—and the target device. Note that the Address variable does not include the Xway address elements {Network.Station} because we are not bridging through another PLC station. Use the ADDR function to convert the following character string to an address:

ADDR(‘0.4.0.10’), where:

- rack = 0
- module (slot number) = 4

- channel = 0
- destination address (target device number) = 10

Configuring the Request_Code Variable

The Request_Code variable identifies the function type for the `SEND_REQ` function block—in this case, a CIP request:

Variable	Description	Value (hex)
Request_Code	Code identifies a CIP request	16#000E

Configuring the Data_to_Send Variable

The Data_to_Send variable identifies the type of explicit message and the CIP request:

Variable	Description	Value (hex)
Data_to_Send[0]	Message type: • 0000 (unconnected), or • 0001 (connected) In this example, unconnected is selected.	16#0000
Data_to_Send[1]	High byte = Request path size (03) Low byte = Service: Get_Attribute_Single (0E)	16#030E
Data_to_Send[2]	High byte = Class (01) Low byte = Class Segment (20)	16#0120
Data_to_Send[3]	High byte = Instance (01) Low byte = Instance Segment (24)	16#0124
Data_to_Send[4]	High byte = Attribute (01) Low byte = Attribute Segment (30)	16#0130

Configuring the Management_Param Variable

The Management_Param variable manages the explicit message:

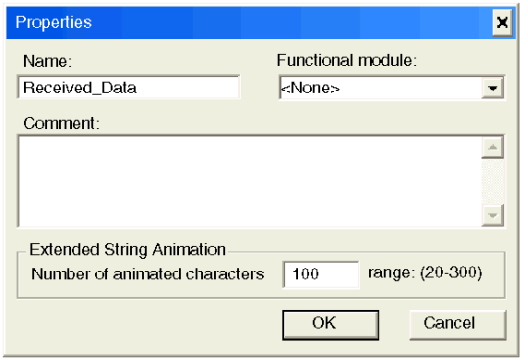
Variable	Description	Value (hex)
Management_Param[0]	High byte = Exchange number (managed by system) Low byte = Activity bit (managed by system)	(read-only)
Management_Param[1]	High byte = Operation report Low byte = Communication report	(read-only)

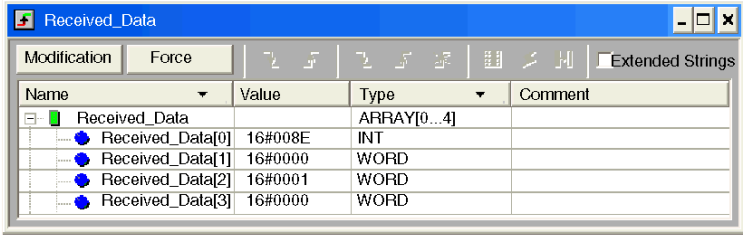
Variable	Description	Value (hex)
Management_Param[2]	Timeout in ms—0 indicates infinite	16#0000
Management_Param[3]	At input = Length of Data_to_Send variable (in bytes) At output = Length of Received_Data variable (in bytes)	16#000A

Viewing the Response

Use a Unity Pro Animation table to display the Received_Data variable array. Note that the Received_Data variable array consists of the entire data buffer.

To display the CIP response, follow these steps:

Step	Action	
1	In Unity Pro, select **Tools** →**Project Browser** to open the Project Browser.	
2	In the Project Browser, select the **Animation Tables** folder, then click the right mouse button. A pop-up menu appears.	
3	Select **New Animation Table** in the pop-up menu. A new animation table and its properties dialog both open.	
4	In the Properties dialog, edit the following values:	
	Name	Type in a table name. For this example: **Received_Data**.
	Functional module	Accept the default **<None>**.
	Comment	(Optional) Type your comment here.
	Number of animated characters	Type in **100**, representing the size of the data buffer in words.
5	The completed **Properties** dialog looks like this:  Click **OK** to close the dialog.	

Step	Action
6	In the animation table's **Name** column, type in the name of the variable assigned to the databuffer: **Received_Data** and hit **Enter**. The animation table displays the Received_Data variable.
7	Expand the Received_Data variable to display its word array, where you can view the CIP response at Received_Data(0-4):  **Note:** Each array entry presents 2 bytes of data in little endian format, where the least significant byte is stored in the smallest memory address. For example, '8E' in word[0] is the lower byte, and '00' is the upper byte.

SEND_REQ Example - Reset

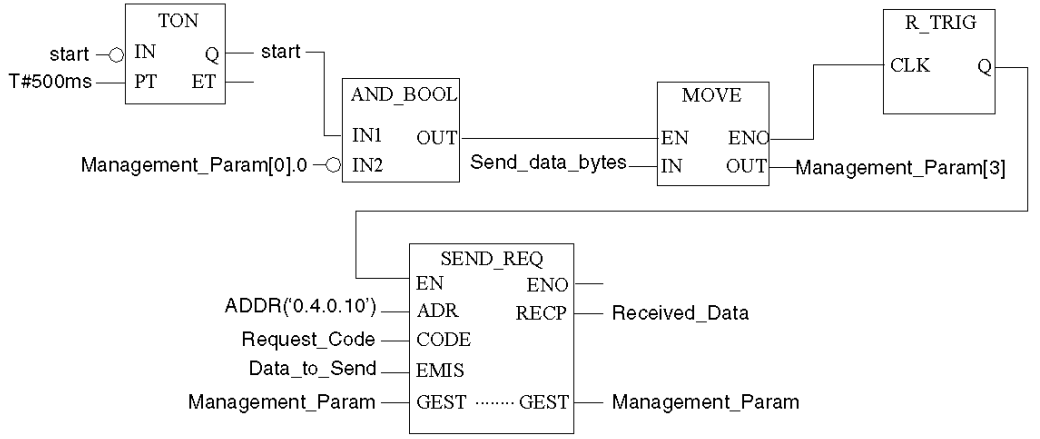
Overview

The following unconnected explicit messaging example shows you how to use the SEND_REQ function block to perform a warm reboot of a third-party device—in this case a Rockwell Automation 1734-AENT PointIO EtherNet/IP adapter (revision 2.1)—using the Reset service.

You can perform the same explicit messaging service using the *Online Action* window of the Unity Pro EtherNet/IP configuration tool *(see page 171)*.

Implementing the SEND_REQ Function Block

To implement the SEND_REQ function block, you need to create and assign variables to the following blocks:



Input Variables

Variables need to be created and assigned to input pins. For the purpose of this example, variables have been created—and named—as described below. (You can, of course, use different variable names in your explicit messaging configurations.)

Input pin	Variable	Data type
IN	start	BOOL
IN	Send_data_bytes	INT
CODE	Request_Code	INT
EMIS	Data_to_Send	Array [0...3] of 4 INT

Note: The value of Send_data_bytes equals the length of the Data_to_Send variable, in bytes. In this example, Send_data_bytes = 16#000A.

Input/Output Variables

A single variable needs to be created and assigned to the dual input/output GEST pins. For the purpose of this example, a variable has been created—and named—as described below. (You can, of course, use different variable names in your explicit messaging configurations.)

Input pin	Variable	Data type
GEST	Management_Param	Array [0...3] of 4 INT

Output Variables

A variable also needs to be created and assigned to the single RECP output pin. (The names assigned to the output variable apply only to this example, and can be changed in your explicit messaging configurations.)

Output pin	Variable	Data type
RECP	Received_Data	Array [0...1] of 2 INT

Configuring the Address Variable

The Address variable identifies the explicit message originating device—-in this example, the TSX ETC 100 communication module—and the target device. Note that the Address variable does not include the Xway address elements {Network.Station} because we are not bridging through another PLC station. Use the ADDR function to convert the following character string to an address:

ADDR('0.4.0.10'), where:

- rack = 0
- module (slot number) = 4

- channel = 0
- destination address (target device number) = 10

Configuring the Request_Code Variable

The Request_Code variable identifies the function type for the `SEND_REQ` function block—in this case, a CIP request:

Variable	Description	Value (hex)
Request_Code	Code identifies a CIP request	16#000E

Configuring the Data_to_Send Variable

The Data_to_Send variable identifies the type of explicit message and the CIP request:

Variable	Description	Value (hex)
Data_to_Send[0]	Message type: • 0000 (unconnected), or • 0001 (connected) In this example, unconnected is selected.	16#0000
Data_to_Send[1]	High byte = Request path size (02) Low byte = Service: Reset (05)	16#0205
Data_to_Send[2]	High byte = Class (01) Low byte = Class Segment (20)	16#0120
Data_to_Send[3]	High byte = Instance (01) Low byte = Instance Segment (24)	16#0124

Configuring the Management_Param Variable

The Management_Param variable manages the explicit message:

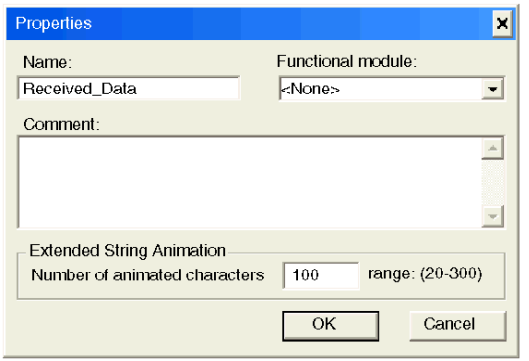
Variable	Description	Value (hex)
Management_Param[0]	High byte = Exchange number (managed by system) Low byte = Activity bit (managed by system)	(read-only)
Management_Param[1]	High byte = Operation report Low byte = Communication report	(read-only)
Management_Param[2]	Timeout in ms—0 indicates infinite	16#0000
Management_Param[3]	At input = Length of Data_to_Send variable (in bytes) At output = Length of Received_Data variable (in bytes)	16#0008

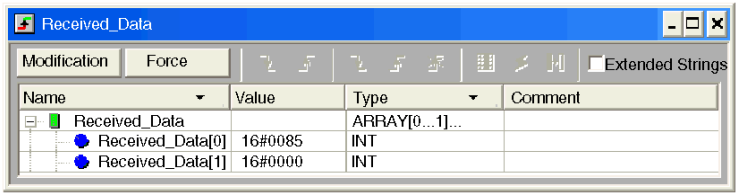
Viewing the Response

Use a Unity Pro Animation table to display the Received_Data variable array. Because the Reset explicit messaging command returns no data, the Received_Data variable includes no CIP response component. The Received_Data variable array includes only the:

- CIP request located in Received_Data(0)
- CIP request status located in Received_Data(1)

To display the contents of the Received_Data variable array, follow these steps:

Step	Action	
1	In Unity Pro, select **Tools →Project Browser** to open the Project Browser.	
2	In the Project Browser, select the **Animation Tables** folder, then click the right mouse button. A pop-up menu appears.	
3	Select **New Animation Table** in the pop-up menu. A new animation table and its properties dialog both open.	
4	In the Properties dialog, edit the following values:	
	Name	Type in a table name. For this example: **Received_Data**.
	Functional module	Accept the default **<None>**.
	Comment	Leave blank.
	Number of animated characters	Type in **100**, representing the size of the data buffer in words.
5	The completed Properties dialog looks like this:  Click **OK** to close the dialog.	

Step	Action
6	In the animation table's *Name* column, type in the name of the variable assigned to the databuffer: **Received_Data** and hit **Enter**. The animation table displays the Received_Data variable.
7	Expand the Received_Data variable to display its word array, where you can view the CIP response at words Received_Data(7-16):  **Note:** Each integer presents 2 bytes of data in little endian format, where the least significant byte is stored in the smallest memory address. For example, '85' in Received_Data[0] is the lower byte, and '00' is the upper byte.

Explicit Messaging - Online Action: Get_Attributes_Single

Overview

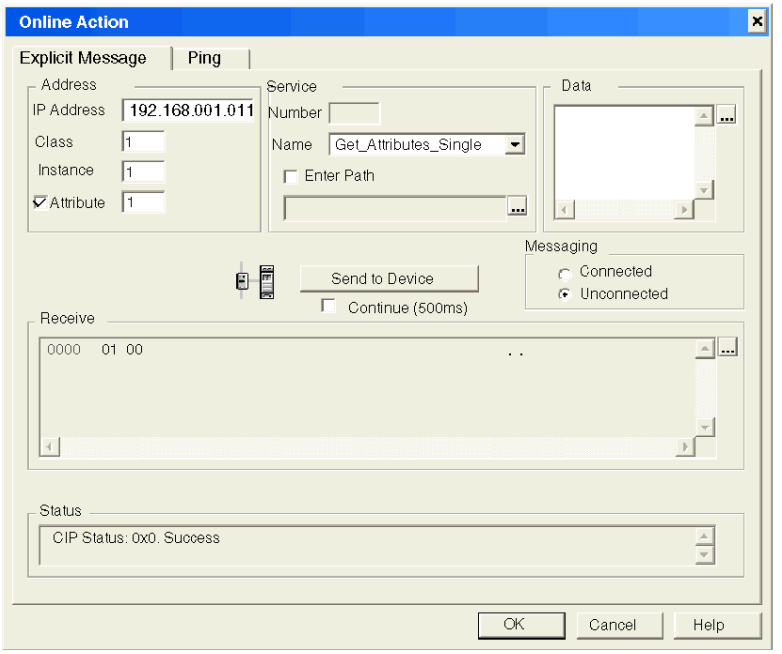
The following example shows you how to use the **Online Action** window in the Unity Pro EtherNet/IP configuration tool to execute an unconnected explicit message that retrieves the vendor ID from a third-party device—in this case a Rockwell Automation 1734-AENT PointIO adapter (revision 2.1)—using the Get_Attributes_Single service.

You can perform the same explicit messaging service using the SEND_REQ function block *(see page 159)*.

Configuring the Explicit Message

To configure, then execute, an unconnected explicit message that will retrieve the vendor ID from a third-party device, follow these steps:

Step	Action
1	Launch the EtherNet/IP configuration tool from the **Configuration** page of the communication module's **Properties** window.
2	In the EtherNet/IP configuration tool, begin on-line operations by clicking the **Go Online** button .
3	Click on the **Network Detection** tab to enable online actions 
4	Open the **Online Action** window by selecting **Network** →**Online Action**.

Step	Action	
5	In the **Explicit Messaging** page, complete the following fields:	
	IP Address	Type in the IP address of the third-party device. In this example, the IP address is: **192.168.001.011**.
	Class	Type in the number that identifies the object class. In this example, the number representing the assembly class object is **1**.
	Instance	Type in the number that identifies the instance of the identity class object. In this example, the number is **1**.
	Attribute	Place a check mark in the checkbox to enable this field, then type in the number identifying the attribute containing vendor ID. In this example, the number is **1**.
	Name	Select the name of the explicit messaging service. In this example, select **Get_Attributes_Single**.
	Messaging	Select the type of explicit message. In this example, select **Unconnected**.
	(The explicit messaging configuration is displayed, below.)	
6	To execute the unconnected explicit message, click **Send to Device**.	
7	The **Receive** area displays the message output, and the **Status** area displays the success or failure of the explicit messaging procedure: 	
8	Click **OK** to close the window.	

Explicit Messaging - Online Action: Reset

Overview

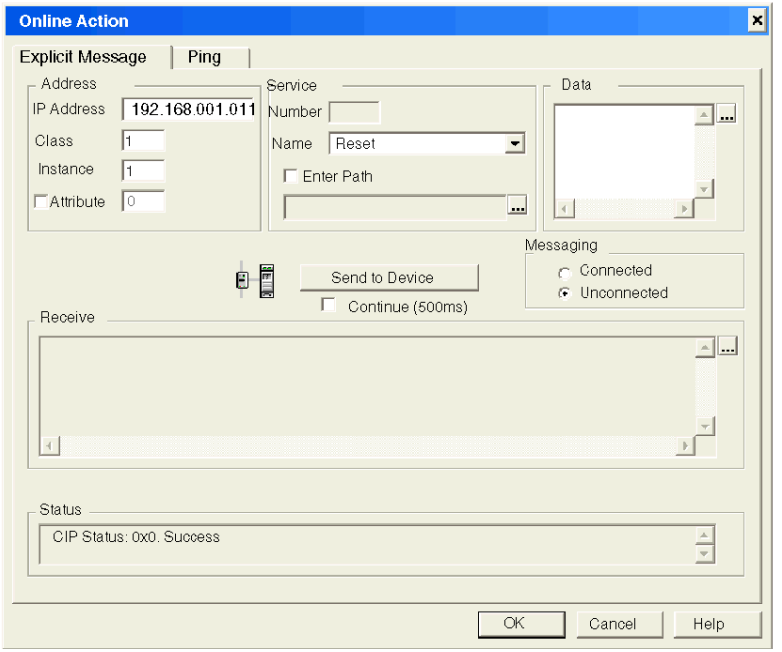
The following example shows you how to use the **Online Action** window in the Unity Pro EtherNet/IP configuration tool to execute an unconnected explicit message that performs a warm reset of a third-party device—in this case the Rockwell Automation 1734-AENT PointIO adapter—using the Reset service.

You can perform the same explicit messaging service using the SEND_REQ function block *(see page 164)*.

Configuring the Explicit Message

To configure, then execute, an unconnected explicit message that will perform a warm reset of a third-party device, follow these steps:

Step	Action	
1	Launch the EtherNet/IP configuration tool from the **Configuration** page of the communication module's **Properties** window.	
2	In the EtherNet/IP configuration tool, begin on-line operations by clicking the **Go Online** button .	
3	Click on the **Network Detection** tab to enable online actions 	
4	Open the **Online Action** window by selecting **Network** →**Online Action**.	
5	In the Explicit Messaging page, complete the following fields:	
	IP Address	Type in the IP address of the STB NIC 2212. In this example, the IP address is: **192.168.001.011**.
	Class	Type in the number that identifies the object class. In this example, the number representing the assembly class object is **1**.
	Instance	Type in the number that identifies the instance of the class object. In this example, the number is **1**.
	Attribute	Do not select this checkbox. The Reset service operates at the instance level, not the attribute level.
	Name	Select the name of the explicit messaging service. In this example, select **Reset**.
	Messaging	Select the type of explicit message. In this example, select **Unconnected**.
	(The explicit messaging configuration is displayed, below.)	

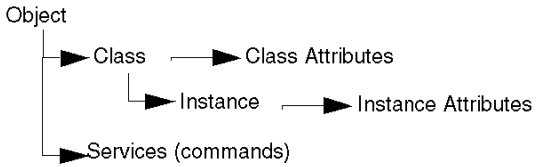
Step	Action
6	To execute the unconnected explicit message, click **Send to Device**.
7	The **Status** area displays the success or failure of the explicit messaging procedure:  **Note**: Because the service returns no data, the **Receive** area displays no message output.
8	Click **OK** to close the window.

CIP Objects

6

Overview

The EtherNet/IP communication module stores data and offers services in a CIP object hierarchy, consisting of the following nested levels:



When the module's local slave service is activated, remote devices can send explicit messages to the module's object hierarchy and perform services that:

- access module data, or
- execute module commands

The local slave function is activated by selecting **Active Configuration** in the **General Configuration** page of the **Local Slave** window *(see page 62)*.

This chapter describes the CIP objects the EtherNet/IP communication module can expose to remote devices.

What Is in This Chapter?

This chapter contains the following topics:

Adapter Diagnostic Object

Overview

The Adapter Diagnostic CIP object consists of the attributes and services described below.

Attributes

The Adapter Diagnostic CIP object consists of the following attributes:

1. Class attributes:

ID (hex)	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported — = not supported			

2. Instance attributes:

ID (hex)	Description	Type	GET	SET	Value
01	Control Bits	WORD	X	X	0: Deactivate checking time for production and consumption (default) 1: Activate
02	ST_DIAG_CNT	STRUCT	X	X	
	wErrFrameCnt	UINT			Incremented each time a frame isn't sent by missing resources or is impossible to send.
	wErrTimeOutCnt	UINT			Incremented when a connection is timed out.
	wErrRefusedCnt	UINT			Incremented when a connection is refused by the remote station.
	dwErrProdCnt	UDINT			Incremented at each production.
	dwErrConsCnt	UDINT			Incremented at each consumption.
	dwErrProdByteCnt	UDINT			Total bytes produced.
	dwErrConsByteCnt	UDINT			Total bytes consumed.
03	Input Status	WORD	X	—	See Status descriptions, below.
04	Output Status	WORD	X	—	See Status descriptions, below.
X = supported — = not supported					

ID (hex)	Description	Type	GET	SET	Value
05	ST_LINK	STRUCT	X	—	
	CIP Status	UINT			See Status descriptions, below.
	Extended Status	UINT			See Status descriptions, below.
	Production Connection ID	DWORD			Connection ID
	Consumed Connection ID	DWORD			Connection ID
	OtoT API	UDINT			API of the Connection
	TtoO API	UDINT			API of the Connection
	OtoT RPI	UDINT			RPI of the Connection
	TtoO RPI	UDINT			RPI of the Connection
06	ST_SOCK_PARAM	STRUCT	X	—	
	lpSockId	DWORD			Internal identifier
	lpForeign	DWORD			IP of the remote station
	wPortForeign	UINT			Port number of the remote station
	lpLocal	DWORD			IP of the local station
	wPortLocal	UINT			Port number of the local station
07	ST_PRODUCTION	STRUCT	X	—	
	bValid	WORD			0: data of the struct production is not valid 1: data of the struct production is valid
	dwCurrentTime	UDINT			(Internal Use—number of ticks before next production)
	dwProductionTime	UDINT			(Internal Use—number of ticks between production)
	SequenceNumber	UDINT			Number of the dwquence in the production
	stCheckTime	STRUCT			
	dwLastTime	UDINT			(Internal Use)
	dwMaxTime	UDINT			Maximum time between 2 productions
	dwMinTime	UDINT			Minimum time between 2 productions
	dwRPI	UDINT			API of the connection
	wOverRun	UINT			Number of times the production was too long
	wUnderRun	UINT			Number of times the production was too short
	dwCurrentTime	UDINT			(Internal Use)
X = supported — = not supported					

ID (hex)	Description	Type	GET	SET	Value
08	ST_CONSUMPTION	STRUCT	X	—	
	bValid	WORD			0: data of the struct consumption is not valid 1: data of the struct consumption is valid
	dwCurrentTime	UDINT			(Internal Use—number of ticks before the timeout)
	dwConsumptionTime	UDINT			(Internal Use—number of ticks in the timeout)
	SequenceNumber	UDINT			Number of the sequence in the consumption
	stCheckTime	STRUCT			
	dwLastTime	UDINT			(Internal Use)
	dwMaxTime	UDINT			Maximum time between 2 consumptions
	dwMinTime	UDINT			Minimum time between 2 consumptions
	dwRPI	UDINT			API of the connection
	wOverRun	UINT			Number of times the consumption was too long
	wUnderRun	UINT			Number of times the consumption was too short
	dwCurrentTime	UDINT			(Internal Use)
09	Connection Entry List	STRUCT	X	—	Status of the CCO object. See Status descriptions, below.
	byGeneralStatus	BYTE			
	byReserved	BYTE			
	Extended Status	WORD			
X = supported — = not supported					

Adapter Status

Adapter status values include the followng:

Status	Description	CIP Status	Extended	Explanation
0	OK	0	0	The I/O data are correctly exchanged.
33	No connection	0	0	No connection.
		0xFB	0xFB01	Connection in timeout.
		0xFB	0xFB07	Optimization error / unknown MAC Address.
		0xFB	0xFB0B	Timeout on consumption.
		0xFB	0xFB0C	Connection closed by a Fw_Close.
		0xFB	0xFB0E	Module in STOP.
		0xFD		Error from encapsulation layer.
		0xFE		Error on TCP connection.
		0x02	0	No more resources to handle the connection.
		0x20	0	Connections refused by bad format or parameters.
53	IDLE	0	0	An IDLE notification is received.
54	Connection in progress	0	0	The connection is established, but I/O data is not yet consumed.

Services

The CIP Adapter Diagnostic object performs the following services upon the listed object types:

ID (hex)	Description	Class	Instance	Notes		
01	Get_Attributes_All	X	X			
61	Get_Output	—	X	Returns the status and values of the output:		
				Offset	Type	Description
				0	UINT	Status
				2	USINT[0...409]	Output Data
62	Get_Intput	—	X	Returns the status and values of the input:		
				Offset	Type	Description
				0	UINT	Status
				2	USINT[0...409]	Input Data
63	Set_DiagCounters	—	X	Sets the values of the structure: • ST_DIAG_CNT to 0, and • ST_CHECK_TIME (production and consumption) to 0 (but not fields dwLastTime and dwCurrentTime) the structure ST_DIAG_CNT to 0.		
X = supported — = not supported						

Assembly Object

Overview

The Assembly CIP object consists of the attributes and services described below.

Attributes

The Assembly CIP object consists of the following attributes:

1. Class attributes:

ID (hex)	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported — = not supported			

2. Instance attributes:

ID (hex)	Description	Type	GET	SET
03	Data	Array of BYTE	X	X
X = supported — = not supported				

Services

The CIP Assembly object performs the following services upon the listed object types:

ID (hex)	Description	Class	Instance	Notes
01	Get_Attributes_All	X	X	—
0E	Get_Attributes_Single	X	X	—
10	Set_Attributes_Single	—	X	Returns these values: 0E=attribute not settable: assembly is not o->T type 0F=permission denied: assembly is being used by an active connection 13=config too small: the Set_Attributes_Single command contains partial data 15=too big data: the Set_Attributes_Single command contains too much data
X = supported — = not supported				

Connection Manager Object

Overview

The Connection Manager CIP object consists of the attributes and services described below.

Attributes

The Connection Manager CIP object consists of the following attributes:

1. Class attributes:

ID (hex)	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported — = not supported			

2. Instance attributes:

ID (hex)	Description	Type	GET	SET	Value
01	Open Requests	UINT	X	X	Number of Forward Open service requests received
02	Open Format Rejects	UINT	X	X	Number of Forward Open service requests that were rejected due to bad format
03	Open Resource Rejects	UINT	X	X	Number of Forward Open service requests that were rejected due to lack of resources
04	Open Other Rejects	UINT	X	X	Number of Forward Open service requests that were rejected for reasons other than bad format or lack of resources
05	Close Requests	UINT	X	X	Number of Forward Close service requests received
X = supported — = not supported					

ID (hex)	Description	Type	GET	SET	Value
06	Close Format Requests	UINT	X	X	Number of Forward Close service requests that were rejected due to bad format
07	Close Other Requests	UINT	X	X	Number of Forward Close service requests that were rejected for reasons other than bad format
08	Connection Timeouts	UINT	X	X	Total number of connection timeouts that occurred in connections controlled by this connections manager
09	Connection Entry List	STRUCT	X	—	List of connections—always 0
11	CPU_Utilization	UINT	X	—	CPU Utilization in tenths of a percent—always 0
12	MaxBuffSize	UDINT	X	—	Amount of buffer space originally available—always 0
13	BufSize Remaining	UDINT	X	—	Amount of buffer space now available—always 0
X = supported — = not supported					

Services

The CIP Connection Manager object performs the following services upon the listed object types:

ID (hex)	Description	Class	Instance	Notes
01	Get_Attributes_All	X	X	—
0E	Get_Attributes_Single	X	X	—
4E	Forward Close	—	X	Managed internally by the EtherNet/IP stack—no link to CPU exists
52	Unconnected Send	—	X	
54	Forward Open	—	X	
X = supported — = not supported				

Ethernet Link Object

Overview

The Ethernet Link CIP object consists of the attributes and services described below.

Attributes

The Ethernet Link CIP object consists of the following attributes:

1. Class attributes:

ID (hex)	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported — = not supported			

2. Instance attributes:

ID (hex)	Description	Type	GET	SET	Value
01	Interface Speed	UDINT	X	—	Valid values include: 0, 10000000, 100000000
02	Interface Flags	DWORD	X	—	Bit 0: Link Status 0 = Inactive 1 = Active
					Bit 1: Duplex Mode 0 = half duplex 1 = full duplex
					Bits 2-4: Negotiation Status 3 = successfully negotiated speed and duplex 4 = forced speed and link
					Bit 5: Manual Setting Requires Reset 0 = automatic 1 = device need reset
					Bit 6: Local Hardware Fault 0 = no fault 1 = fault detected
X = supported — = not supported					

ID (hex)	Description	Type	GET	SET	Value
03	Physical Address	ARRAY of 6 USINT	X	—	Module MAC Address
04	Interface Counters	STRUCT	X	—	
	In octets	UDINT			Octets received on the interface
	In Ucast Packets	UDINT			Unicast packets received on the interface
	In NUcast Packets	UDINT			Non-unicast packets received on the interface
	In Discards	UDINT			Inbound packets received on the interface, but discarded
	In Errors	UDINT			Inbound packets that contain errors (does not include In Discards)
	In Unknown Protos	UDINT			Inbound packets with unknown protocol
	Out Octets	UDINT			Octets sent on the interface
	Out Ucast Packets	UDINT			Unicast packets sent on the interface
	Out NUcast Packets	UDINT			Non-unicast packets sent on the interface
	Out Discards	UDINT			Outbound packets discarded
	Out Errors	UDINT			Outbound packets that contain errors
X = supported — = not supported					

ID (hex)	Description	Type	GET	SET	Value
05	Media Counters	STRUCT	X	—	
	Alignment Errors	UDINT			Frames that are not an integral number of octets in length
	FCS Errors	UDINT			Frames received that do not pass the FCS check
	Single Collisions	UDINT			Successfully transmitted frames that experienced exactly one collision
	Multiple Collisions	UDINT			Successfully transmitted frames that experienced more than one collision
	SQE Test Errors	UDINT			Number of times the SQE test error is generated
	Deferred Transmissions	UDINT			Frames for which first transmission attempt is delayed because the medium is busy
	Late Collisions	UDINT			Number of times a collision is detected later than 512 bittimes into the transmission of a packet
	Excessive Collisions	UDINT			Frames for which transmission fails due to excessive collisions
	MAC Transmit Errors	UDINT			Frames for which transmission fails due to internal MAC sublayer transmit error
	Carrier Sense Errors	UDINT			Times that the carrier sense condition was lost or never asserted when attempting to transmit a frame
	Frame Too Long	UDINT			Frames received that exceed the maximum permitted frame size
	MAC Receive Errors	UDINT			Frames for which reception on an interface fails due to an internal MAC sublayer receive error

X = supported
— = not supported

ID (hex)	Description	Type	GET	SET	Value
06	Interface Control	STRUCT	X	X	API of the connection
	Control Bits	WORD			Bit 0: Auto-negotiation 0 = Disabled 1 = Enabled **Note:** When auto-negotiation is enabled, the error 0x0C (Object State Conflict) is returned when attempting to set either: • Forced Interface Speed, or • Forced Duplex Mode
					Bit 1: Forced Duplex Mode (if auto-negotiation bit = 0) 0 = half duplex 1 = full duplex
	Forced Interface Speed	UINT			Valid values include: 10000000, 100000000 **Note:** Attempting to set any other value returns the error 0x09 (Invalid Attribute Value)
X = supported — = not supported					

Services

The CIP Ethernet Link object performs the following services upon the listed object types:

ID (hex)	Description	Class	Instance
01	Get_Attributes_All	X	X
05	Set_Attribute_Single	—	X
0E	Get_Attribute_Single	X	X
X = supported — = not supported			

Identity Object

Overview

The Identity CIP object consists of the attributes and services described below.

Attributes

The Identity CIP object consists of the following attributes:

1. Class attributes:

ID (hex)	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported — = not supported			

2. Instance attributes:

ID (hex)	Description	Type	GET	SET
01	Vendor ID	UINT	X	—
02	Device Type	UINT	X	—
03	Product Code	UINT	X	—
04	Revision	STRUCT	X	—
	Major	USINT		
	Minor	USINT		
05	Status bit 2: 0x01=the module is configured bits 4-7: 0x03=no I/O connections established 0x06=at least 1 I/O connection in run mode 0x07=at least 1 I/O connection established, all in IDLE mode	Word	X	—
06	Serial Number	UDINT	X	—
07	Product Name	STRING	X	—
X = supported — = not supported				

Services

The CIP Identity object performs the following services upon the listed object types:

ID (hex)	Description	Class	Instance	Notes
01	Get_Attributes_All	X	X	Applies to all class and all instance attributes
0E	Get_Attributes_Single	X	X	Applies to all class and all instance attributes
05	Reset	—	X	Two types: 00=power cycle 01=return to factory defaults and power cycle
X = supported — = not supported				

Module Diagnostic Object

Overview

The Module Diagnostic CIP object consists of the attributes and services described below.

Attributes

The Module Diagnostic CIP object consists of the following attributes:

1. Class attributes:

ID (hex)	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported — = not supported			

2. Instance attributes:

ID (hex)	Description	Type	GET	SET	Value
01	Module Status	WORD	X	—	01=started 02=stopped 03=running
02	CNF Version	WORD	X	—	0x0100
03	CRC	UDINT	X	—	—
04	I/O Connection Status	STRUCT	X	—	—
	Size Table	WORD			size (16 bytes)
	Table	WORD[]			table of I/O status (8 WORDS) 1=INPUT and OUTPUT status of I/O connection are OK 0=at least 1 INPUT or OUTPUT status of I/O connection is not OK
05	Cco Mode	WORD	X	X	01=activate status to CCO in the module 02=block access to CCO
X = supported — = not supported					

Services

The CIP Module Diagnostic object performs the following services upon the listed object types:

ID (hex)	Description	Class	Instance	Notes
01	Get_Attributes_All	X	X	—
0E	Set_Attributes_Single	—	X	—
X = supported — = not supported				

Scanner Diagnostic Object

Overview

The Scanner Diagnostic CIP object consists of the attributes and services described below.

Attributes

The Scanner Diagnostic CIP object consists of the following attributes:

1. Class attributes:

ID (hex)	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported — = not supported			

2. Instance attributes:

ID (hex)	Description	Type	GET	SET	Value
01	Control Bits	WORD	X	X	0: Deactivate checking time for production and consumption (default) 1: Activate
02	ST_DIAG_CNT	STRUCT	X	X	
	wErrFrameCnt	UINT			Incremented each time a frame isn't sent by missing resources or is impossible to send.
	wErrTimeOutCnt	UINT			Incremented when a connection is timed out.
	wErrRefusedCnt	UINT			Incremented when a connection is refused by the remote station.
	dwErrProdCnt	UDINT			Incremented at each production.
	dwErrConsCnt	UDINT			Incremented at each consumption.
	dwErrProdByteCnt	UDINT			Total bytes produced.
	dwErrConsByteCnt	UDINT			Total bytes consumed.
03	Input Status	WORD	X	—	See Status descriptions, below.
04	Output Status	WORD	X	—	See Status descriptions, below.
X = supported — = not supported					

ID (hex)	Description	Type	GET	SET	Value
05	ST_LINK	STRUCT	X	—	
	CIP Status	UINT			See Status descriptions, below.
	Extended Status	UINT			See Status descriptions, below.
	Production Connection ID	DWORD			Connection ID
	Consumed Connection ID	DWORD			Connection ID
	OtoT API	UDINT			API of the Connection
	TtoO API	UDINT			API of the Connection
	OtoT RPI	UDINT			RPI of the Connection
	TtoO RPI	UDINT			RPI of the Connection
06	ST_SOCK_PARAM	STRUCT	X	—	
	lpSockId	DWORD			Internal identifier
	lpForeign	DWORD			IP of the remote station
	wPortForeign	UINT			Port number of the remote station
	lpLocal	DWORD			IP of the local station
	wPortLocal	UINT			Port number of the local station
07	ST_PRODUCTION	STRUCT	X	—	
	bValid	WORD			0: data of the struct production is not valid 1: data of the struct production is valid
	dwCurrentTime	UDINT			(Internal Use—number of ticks before next production)
	dwProductionTime	UDINT			(Internal Use—number of ticks between production)
	SequenceNumber	UDINT			Number of the dwquence in the production
	stCheckTime	STRUCT			
	dwLastTime	UDINT			(Internal Use)
	dwMaxTime	UDINT			Maximum time between 2 productions
	dwMinTime	UDINT			Minimum time between 2 productions
	dwRPI	UDINT			API of the connection
	wOverRun	UINT			Number of times the production was too long
	wUnderRun	UINT			Number of times the production was too short
	dwCurrentTime	UDINT			(Internal Use)

X = supported
— = not supported

ID (hex)	Description	Type	GET	SET	Value
08	ST_CONSUMPTION	STRUCT	X	—	
	bValid	WORD			0: data of the struct consumption is not valid 1: data of the struct consumption is valid
	dwCurrentTime	UDINT			(Internal Use—number of ticks before timeout)
	dwConsumptionTime	UDINT			(Internal Use—number of ticks of the timeout)
	SequenceNumber	UDINT			Number of the sequence in the consumption
	stCheckTime	STRUCT			
	dwLastTime	UDINT			(Internal Use)
	dwMaxTime	UDINT			Maximum time between 2 consumptions
	dwMinTime	UDINT			Minimum time between 2 consumptions
	dwRPI	UDINT			API of the connection
	wOverRun	UINT			Number of times the consumption was too long
	wUnderRun	UINT			Number of times the consumption was too short
	dwCurrentTime	UDINT			(Internal Use)
09	Connection Entry List	STRUCT	X	—	Status of the CCO object. See Status descriptions, below.
	byGeneralStatus	BYTE			
	byReserved	BYTE			
	Extended Status	WORD			
X = supported — = not supported					

Scanner Status

Scanner status values include the followng:

Status	Description	CIP Status	Extended	Explanation
0	OK	0	0	The I/O data are correctly exchanged.
33	Timeout	0xFB	0xFB0B	Timeout detected on consumption.
53	IDLE	0	0	An IDLE notification is received.
54	Connection established	0	0	The connection is established, but I/O is not yet consumed.
		0xFB	0xFB08	Impossible to start the production.
		0xFB	0xFB09	Impossible to start the consumption.
		0xFB	0xFB0A	Not enough resources to manage the connection.

Status	Description	CIP Status	Extended	Explanation
58	Not connected (TCP)	0xFE		Error on TCP connection.
65	Not connected (CIP)	0xFB	0xFB01	Timeout for Fw_Open response.
		0xFB	0xFB02	Bad format of the Fw_Open response (so addr).
		0xFB	0xFB03	Bad parameters in the Fw_Open response (OT Net Par).
		0xFB	0xFB04	Bad parameters in the Fw_Open response (TO Net Par).
		0xFB	0xFB05	Fw_Open respons asks for port number other than 2222.
		0xFB	0xFB06	Error joining the UDP multicast group.
		0xFB	0xFB07	Optimization error / unknown MAC Address.
68	Connection establishing	0xD0	0x0001	Connection is closed.
		0xD0	0x0002	Connection is pending.
70	Not connected (EPIC)	0xFD		Error code in register session response.
		0xFD		Error code in the frame.
		0xFD		Encapsulation session un-registered.
77	Scanner stopped	0	0	Connection is stopped.

Services

The CIP Scanner Diagnostic object performs the following services upon the listed object types:

ID (hex)	Description	Class	Instance	Notes		
01	Get_Attributes_All	X	X			
61	Get_Output	—	X	Returns the status and values of the output:		
				Offset	Type	Description
				0	UINT	Status
				2	USINT[0...409]	Output Data
62	Get_Input	—	X	Returns the status and values of the input:		
				Offset	Type	Description
				0	UINT	Status
				2	USINT[0...409]	Input Data
63	Set_DiagCounters	—	X	Sets the value of the structure ST_DIAG_CNT to 0.		

X = supported
— = not supported

TCP/IP Interface Object

Overview

The Identity CIP object consists of the attributes and services described below.

Attributes

The TCP/IP Interface CIP object consists of the following attributes:

1. Class attributes:

ID (hex)	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported — = not supported			

2. Instance attributes:

ID (hex)	Description	Type	GET	SET	Value
01	Status	DWORD	X	—	always = 0x01
02	Configuration Capability	DWORD	X	—	0x01 = from BootP 0x11 = from flash 0x00 = other
03	Configuration Control	DWORD	X	X	0x01 = out-of-box default
04	Physical Link Object	STRUCT	X	—	
	Path Size	UINT			
	Path	Padded EPATH			
05	Interface Configuration	STRUCT	X	X	0x00 = out-of-box default
	IP Address	UDINT			
	Network Mask	UDINT			
	Gateway Address	UDINT			
	Name Server	UDINT			
	Name Server 2	UDINT			
	Domain Name	STRING			
06	Host Name	STRING	X	—	
X = supported — = not supported					

Services

The CIP TCP/IP Interface object performs the following services upon the listed object types:

ID (hex)	Description	Class	Instance	Notes
01	Get_Attributes_All	X	X	—
0E	Get_Attributes_Single	X	X	—
05	Get_Attributes_Single	—	X	—
X = supported — = not supported				

Diagnostics

7

Overview

This chapter describes the diagnostic features of the EtherNet/IP communication module and the Unity Pro EtherNet/IP configuration software.

What Is in This Chapter?

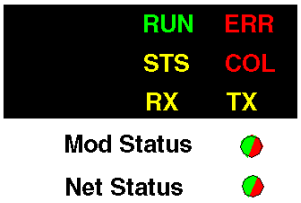
This chapter contains the following sections:

7.1 LED Indicators

LED Indicators for the TSX ETC 100 Module

LED Indicators

The TSX ETC 100 displays the following LED indicators:



LED Descriptions

Use the LED display to diagnose the state of the module, as follows:

LED	Color	Description
RUN	Green	• **Off:** Indicates that the module is not communicating with the CPU over the backplane. • **Steady Green:** Indicates that the module is communicating with the CPU over the backplane.

ERR	Red	• **Off:**
		  ◦ Power is not being supplied to the module.
		  ◦ The module is running a self test.
		  ◦ The module is ready to start operating.
		• **Steady Red:**
		  ◦ The module is not operational.
		  ◦ A software operation error has occurred, causing a re-initialization.
		• **Flashing Red:** The module is not configured or is being configured.

LED	Color	Description

STS	Yellow	• **Off:**
		  • Power is not being supplied to the module.
		  • The module is not ready to start operating.
		  • The module is ready to start operating.
		• **Steady Yellow:**
		  • The module is running a self test.
		  • A software operation error has occurred causing a re-initialization.
		  • The module is configured and operational.
		• **Steady or Flashing Yellow** The module is not configured or is being configured.
		The STS LED also provides diagnostic information by using the following sequence of flashes.
		• **Two flashes:** The module does not have a MAC address.
		• **Three flashes:** The Ethernet link is not connected.
		• **Four flashes:** The module has detected a duplicate IP address.
		• **Five flashes:** The module is configured as a BOOTP client and is waiting for a BOOTP server response.
		• **Six flashes:** The module is using its default IP configuration (the module is not configured.)
		• **Seven flashes:** The module has detected a configuration error.
		Note: If more then one of the above conditions is detected, the module will display the condition with the smaller flashing sequence.

COL	Red	• **Flashing Red:** A collision has been detected on the Ethernet link.
RX (Reception Activity)	Yellow	• **Off:** There is no reception activity. • **Flashes Yellow:** Indicates reception activity.
TX (Transmission Activity)	Yellow	• **Off:** There is no transmission activity. • **Flashes Yellow:** Indicates transmission activity.
Mod Status (Module Status)	Green/ Red	• **Off:** Power is not being supplied to the module. • **Steady Green:** The module is operating normally. • **Flashing Green:** The module has not been configured. • **Steady Red:** The module has detected a major fault. • **Flashing Red:** The module has detected a recoverable fault.
Net Status (Network Status)	Green/ Red	• **Off:** Power is not being supplied to the module or the module does not have an IP address assigned. • **Steady Green:** The module has established at least one CIP connection. • **Flashing Green:** The module has obtained an IP address but has not established any CIP connections. • **Steady Red:** The module has detected that its IP address is a duplicate IP address. • **Flashing Red:** One or more CIP connections has timed out and the connection(s) need to be re-established or the module has been reset.

7.2 Diagnostic Testing Using Unity Pro

Overview

This section describes the diagnostic tools available in the Unity Pro software.

What Is in This Section?

This section contains the following topics:

Accessing the Unity Pro Diagnostic Tools

Overview

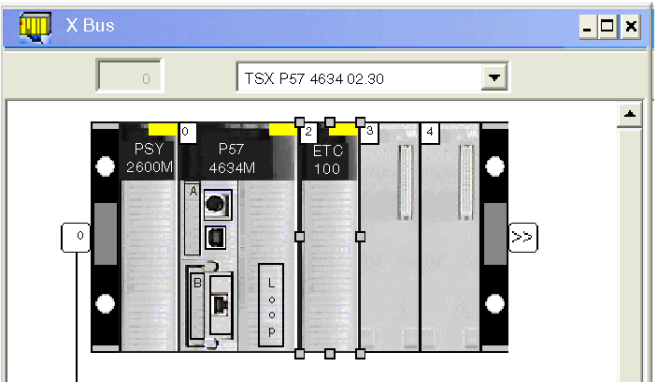
The Unity Pro software provides diagnostic tools that let you view the:

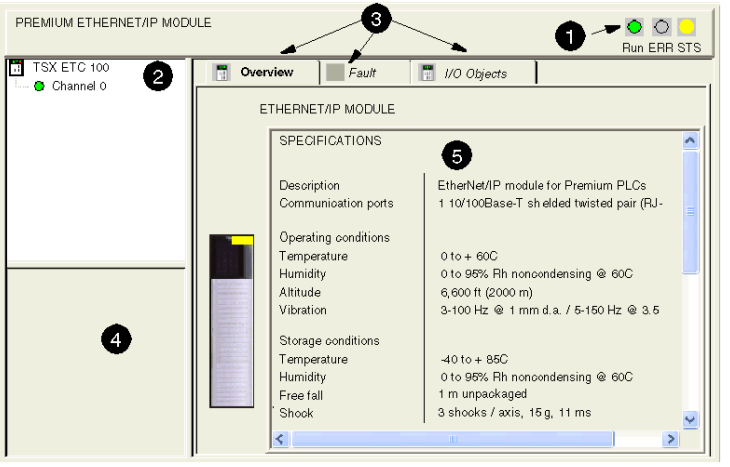
- communication module status
- communication module's:
  - faults
  - I/O objects
- communication channel's:
  - MAC Address
  - IP Address settings
  - faults

These Unity Pro diagnostic tools are available in the Premium EtherNet/IP Module properties window, only when Unity Pro is operating online.

Accessing Unity Pro Diagnostic Tools

To access diagnostic tools for the TSX ETC 100 EtherNet/IP module:

Step	Action
1	Open the Unity Pro project backplane diagram, below: 

Step	Action
2	Do one of the following: ● double click the left mouse button on the TSX ETC 100 module in the window above, or ● click the right mouse button on the TSX ETC 100 module, then select **Open Module...** in the popup menu The Premium EtherNet/IP Module window opens: 

Step	Action	
3	Navigate the Premium EtherNet/IP Module window using the following features:	
	1 Module status icons	These three indicators display the module's status in online mode.
	2 Channel area	Select a node to display parameters for either: • the communication module, or • a communication channel

	3 Page tabs	Select a page to display module or channel properties:
		• for the communication module:
		  • Overview
		  • Fault
		  • I/O Objects
		• for a communication channel:
		  • Configuration
		  • Debug
		  • Fault

	4 General parameters	View communication channel parameters: • **Function** displays the configured communication function and is read-only. • **Task** displays the task (configured **MAST**) and is read-only.
	5 Mode parameters	Displays parameters for the mode you select by opening a page.

Communication Channel Diagnostics in Unity Pro

Overview

Select a communication channel in the **Channel area** to access the:

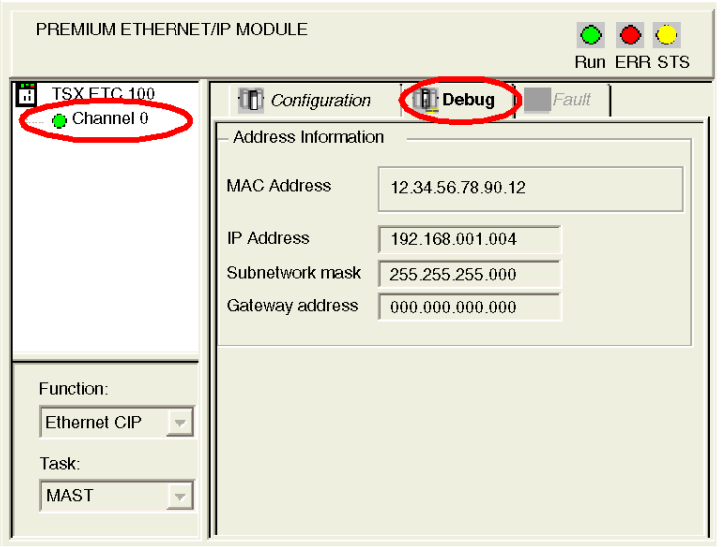
- **Configuration** page, where you can:
  - edit the EtherNet/IP Module name
  - edit input and output data size and location settings
  - launch the Unity Pro EtherNet/IP configuration tool

  Refer to the description of the **Configuration** page *(see page 21)* for more information.
- **Debug** page, which displays the communication module's:
  - MAC Address
  - IP Address settings
- **Fault** page, which displays active faults for the communication channel

MAC Address

To display the MAC Address of the communication module:

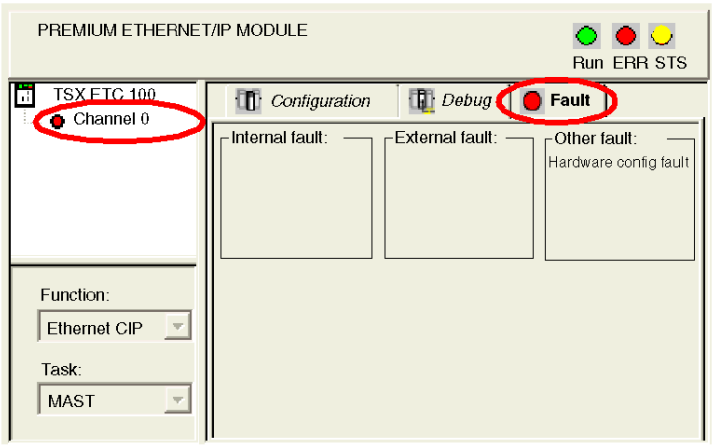
Step	Action
1	In the **Channel area,** select the communication channel. The following pages are displayed: • Configuration • Debug • Default

Step	Action
2	To display the communication module's MAC Address and IP Address settings, click on the **Debug** page: 

Channel Faults

To display the active faults on the communication channel:

Step	Action
1	In the **Channel area,** select the communication channel.

Step	Action
2	To display the communication module's active faults, click on the **Fault** page: 

NOTE: You can also access the channel error bit (CH_ERROR) by using the Unity Pro **Animation Table** to display the **%Ir.m.ch.ERR** object.

Communication Module Diagnostics in Unity Pro

Overview

Use the Premium EtherNet/IP Module window in Unity Pro to diagnose the TSX ETC 100 module. In this window, you can access:

- three icons that reflect the current status of selected LEDs
- the **Overview** page, where you can view a description of the module
- **Fault** page, which displays active faults for the communication module
- **I/O Objects** page, where you can view and manage I/O objects for the module

Module Status Icons

The Premium EtherNet/IP Module window displays three icons that reflect the current status of the following LEDs:

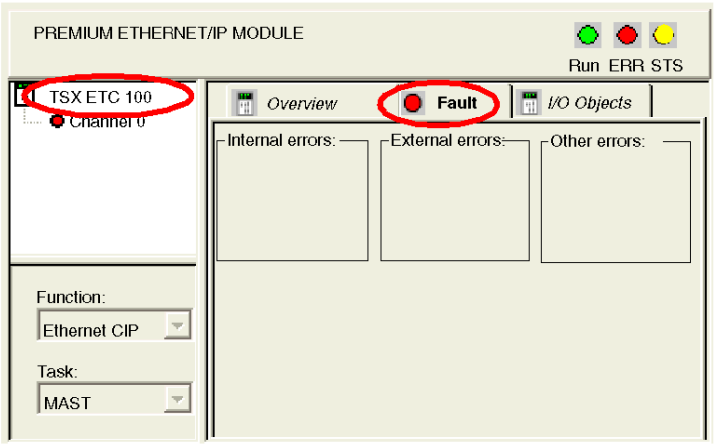
- Run
- ERR
- STS

Refer to the description of LED indicators *(see page 198)* for information on how to use these icons.

Accessing Module Faults

To display the active faults on the communication module:

Step	Action
1	In the **Channel area,** select the communication module. The following pages are displayed: • Overview • Fault • I/O Objects

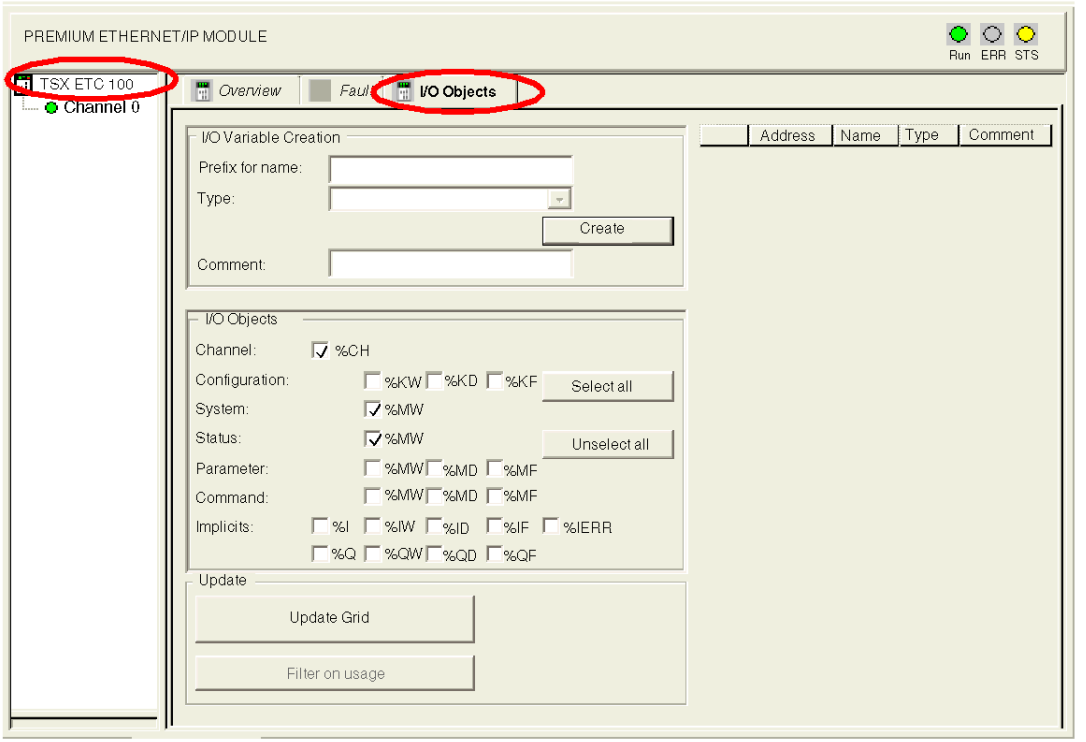
Step	Action
2	To display the communication module's active faults, click on the **Fault** page: 

NOTE: You can also access the module error bit by using the Unity Pro **Animation Table** to display the **%Ir.m.MOD.ERR** object.

Managing I/O Objects

Use the I/O Objects page to view module I/O objects, and to manage the association of these objects with variables.

Open the **I/O Objects** page by selecting the **I/O Objects tab**, after the communication module has been selected in the **Channel area**:



NOTE:

- The TSX ETC 100 EtherNet/IP communication module supports only Channel, System, and Status I/O Objects. Not all bits are used.
- Refer to the Unity Pro help for instructions on how to use the **I/O Objects** page.

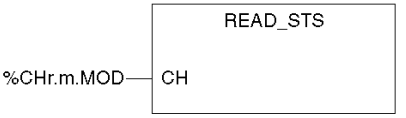
Reading I/O Objects

Use a `READ_STS` function block in Unity Pro to update each of the following types of data:

- module data
- channel data

Updating module data:

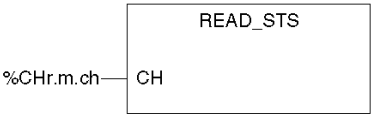
To display module information, follow these steps:

<table>
<tr><th>Step</th><th colspan="2">Action</th></tr>
<tr><td rowspan="4">1</td><td colspan="2">Configure the READ_STS function block, as follows:
 

Where:</td></tr>
<tr><td colspan="2">r = rack, or station, number</td></tr>
<tr><td colspan="2">m = module, or slot, number</td></tr>
<tr><td colspan="2">MOD = a constant indicating module data</td></tr>
<tr><td rowspan="11">2</td><td colspan="2">To view the data updated by the READ_STS function block, enter the corresponding direct addresses in the Unity Pro Animation table, or use them in your program logic:</td></tr>
<tr><td>Object</td><td>Description</td></tr>
<tr><td>%MWr.m.MOD.0</td><td>Exchange Status:
Bit 0: reading of module status in progress</td></tr>
<tr><td>%MWr.m.MOD.1</td><td>Exchange Report:
Bit 0: error while reading module status</td></tr>
<tr><td rowspan="8">%MWr.m.MOD.2</td><td>Bit 0: internal fault</td></tr>
<tr><td>Bit 1: operational fault</td></tr>
<tr><td>Bit 2: not used</td></tr>
<tr><td>Bit 3: self test</td></tr>
<tr><td>Bit 4: not used</td></tr>
<tr><td>Bit 5: configuration fault</td></tr>
<tr><td>Bit 6: missing module or off</td></tr>
<tr><td>Bit 7: not used</td></tr>
</table>

Updating channel data:

To display channel information, follow these steps:

Step	Action	
1	Configure the READ_STS function block, as follows:  Where:	
	r = rack, or station, number	
	m = module, or slot, number	
	ch = channel number—always set to 0 for ETC transactions	
2	To view the data updated by the READ_STS function block, enter the corresponding direct addresses in the Unity Pro Animation table, or use them in your program logic:	
	Object	**Description**
	%MWr.m.ch.0	Exchange Status (EXCH_STS):
		Bit 0: reading of status words of the channel in progress (STS_IN_PROG)
		Bit 15: reconfiguration in progress (RECONF_IN_PROG)
	%MWr.m.ch.1	Exchange Report (EXCH_RPT):
		Bit 0: error while reading channel status (STS_ERR)
		Bit 15: error while reconfiguring the channel (RECONF_ERR)
	%MWr.m.ch.2	Standard channel status (low byte):
		Bits 0...3: reserved (0)
		Bit 4: internal fault
		Bit 5: configuration fault
		Bit 6: X-Bus communication fault
		Bit 7: application fault (conf fault)
		High byte:
		Bits 0...7: reserved (0)

Step	Action	
2 cont'd	%MWr.m.ch.3	Ethernet Port Global Status:
		Bit 0: configuration error
		Bit 1: the Ethernet interface is disabled
		Bit 2: duplicate IP address detected
		Bit 3: reserved
		Bit 4: the Ethernet link is disconnected
		Bit 5: the module is in the process of obtaining an IP address
		Bits 6...15: reserved
	%MWr.m.ch.4	IP address:
		• During normal operation, the double word %MDr.m.c.4 contains the IP address configured or served to the module.
		• In No Configuration state, the double word %MDr.m.c.4 contains the default IP address of the module.
		• In Configuration Error, the double word %MDr.m.c.4 contains the default IP address of the module.
		• When a duplicate IP address is detected, the double word %MDr.m.c.4 contains the served or configured duplicate IP address.
		• When the module is waiting for a BOOTP response, the double word %MDr.m.c.4 contains the IP address 0.0.0.0.

7.3 Diagnostic Testing Using the Unity Pro EtherNet/IP Configuration Tool

Overview

This section describes the diagnostic tools available in the Unity Pro EtherNet/IP configuration tool.

What Is in This Section?

This section contains the following topics:

Diagnostic Testing Using the Unity Pro EtherNet/IP Software

Overview

Use the Unity Pro EtherNet/IP configuration tool to perform a diagnostic test of the EtherNet/IP module and all other devices in your configuration.

NOTE: Diagnostic testing is performed directly between the Unity Pro EtherNet/IP configuration tool software running on your PC and the target EtherNet/IP device.

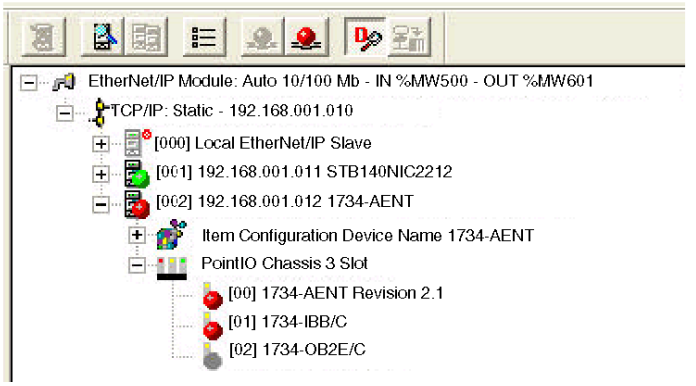
The software displays the results of the diagnostic test, as follows:

- the task bar's **Module State** indicator reads **Diagnostic**
- the **Devices** window depicts the state of connections for each device using a:
  - green icon, indicating all connections are functioning
  - red icon, indicating at least one connection has failed
  - gray icon, indicating a rack optimized module connection
- a diagnostic tab is added to the properties window for each EtherNet/IP device and I/O module displaying:
  - each connection's status, information, and performance data, and
  - the value of each input and output

Performing a Diagnostic Test

To perform a diagnostic test in the Unity Pro EtherNet/IP software:

Step	Action
1	Do one of the following: • click the Diagnostics toolbar button  , or • select **Devices →Diagnostic...** The configuration tool enters its diagnostic state.

Step	Action
2	The EtherNet/IP module enters a diagnostic state and displays the status of each connected device and module. An example of a diagnostic status display appears, below:  In the above example: ● a green icon indicates that all connections are functioning for the device at address [001] ● a red icon indicates at least one connection has failed for the device at address [002] ● red icons indicates that at least one connection has failed for the modules at slots [00] and [01] ● a gray icon indicates a rack optimized connection, exists for the module at slot [02]
3	To exit diagnostic mode, repeat the command in step 2.

Ping a Network Device

Overview

Use the Unity Pro EtherNet/IP configuration tool's Ping function to send an ICMP echo request to a target EtherNet/IP device to determine:

- if the target device is present, and if so
- the elapsed time to receive an echo response from the target device

The target device is identified by its IP address setting. The Unity Pro EtherNet/IP configuration tool will verify that the target address is not a:

- loopback address (127.000.000.000 to 127.255.255.255)
- multicast address (224.000.000.000 to 239.255.255.255)
- reserved address (240.000.000.000 to 255.255.255.255)
- broadcast address

The ping function can be performed from either the:

- **General** page of a device's properties window
- **Ping** page of the **Online Action** window

Pinging a Network Device

To ping a network device:

Step	Action
1	Be sure the Unity Pro EtherNet/IP configuration tool is operating online.
2	Do one of the following: ● Select **Network** →**Online Action**, then click on the **Ping** page, or ● Select a device in the **Devices** window, then select **Devices** →**Properties**
3	If you are working in the Ping page of the Online Action window, type in the **IP Address** of the target device. **Notes**: ● The default is the IP address of the device currently selected in the **Network Detection** list. ● If you are working in the **General** page of a device's **Properties** window, the Unity Pro EtherNet/IP configuration tool uses the IP address of the device selected in the **Devices** window.
4	To send... ● a single ping, de-select the **Loop** checkbox ● a series of pings—1 every 100 ms—select **Loop**
5	(Optional) Select **Stop on Error** to stop pinging if an error occurs.
6	Click **Ping** once to begin pinging.
7	Click **Ping** a second time to stop looped pinging, where no error has been detected.

Viewing Output Messages in the Unity Pro EtherNet/IP Configuration Tool

Overview

Use the Unity Pro EtherNet/IP configuration tool's **Output Message** window to diagnose the health of your EtherNet/IP network. This window maintains a log of network events. You can:

- show or hide the window
- display for each item in the window its:
  - date and time
  - level of significance
- copy the contents of the **Output Message** window to your PC's Windows Clipboard
- clear the contents of the window

Show/Hide the Output Message Window

The **Output Message** window is displayed in the Unity Pro EtherNet/IP configuration tool by default. To hide the window, select:
File →Preferences →Output Window.

To reopen the **Output Message** window, repeat the above command.

Add Date/Time and Level to Output Message Window Items

To show or hide the date and time, or level of significance for **Output Message** window entries:

Step	Action
1	Select **File →Message View →Configuration**. The Output Message View Configuration dialog opens.
2	Select—or de-select—either or both: • **Add Date to Messages** • **Add Level to Messages**
3	Click **OK**.

Copy/Clear

To copy the contents of the **Output Message** window to your PC's Windows Clipboard, select: **File →Message View →Copy**.

To clear the contents of the **Output Message** window select:
File →Message View →Clear.

Replacing the EtherNet/IP Communication Module

8

Replacing the EtherNet/IP Communication Module

Overview

Replacing the module involves removing the old module and mounting a new one in its place.

When to Replace

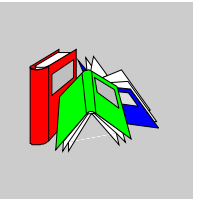
You can replace the EtherNet/IP communication module at any time using another module with compatible firmware. A communication module can be replaced when power to the module is either:

- off (cold swap), or
- on (hot swap)

The replacement module obtains its IP address and operating parameters over the backplane connection from the CPU. The transfer occurs either immediately (hot swap) or when power is next cycled to the device (cold swap).

NOTE: The operating parameters the CPU sends to a replacement module do not include any parameter values that were edited in the original module using explicit messaging "SET" commands. Explicit messaging can be performed in the Online Action window of the Unity Pro EtherNet/IP configuration tool.

Glossary



A

Adapter

An adapter is the target of real-time I/O data connection requests from scanners. It cannot send or receive real-time I/O data unless it is configured to do so by a scanner, and it does not store or originate the data communications parameters necessary to establish the connection. An adapter accepts explicit message requests (connected and unconnected) from other devices.

Advanced mode

In Unity Pro, Advanced mode is a selection that displays expert-level configuration properties that help define Ethernet connections. Because these properties are designed to be edited only by persons with a solid understanding of communication protocols, they can be hidden or displayed, depending upon the qualifications of the specific user.

B

BOOTP

(*bootstrap protocol*) A UDP network protocol that can be used by a network client to automatically obtain an IP address from a server. The client identifies itself to the server using its MAC address. The server—which maintains a pre-configured table of client device MAC addresses and associated IP addresses—sends the client its defined IP address. The BOOTP service utilizes UDP ports 67 and 68.

Broadcast

A message sent to all devices in the subnet.

C

CIP™

(*Common Industrial Protocol*) A comprehensive suite of messages and services for the collection of manufacturing automation applications—control, safety, synchronization, motion, configuration and information. CIP allows users to integrate these manufacturing applications with enterprise-level Ethernet networks and the Internet. CIP is the core protocol of EtherNet/IP.

Class 1 connection

A CIP transport class 1 connection used for I/O data transmission via Implicit Messaging between EtherNet/IP devices.

Class 3 connection

A CIP transport class 3 connection used for Explicit Messaging between EtherNet/IP devices.

connected messaging

In EtherNet/IP, connected messaging uses a CIP connection for communication. A connected message is a relationship between two or more application objects on different nodes. The connection establishes a virtual circuit in advance for a particular purpose, such as frequent explicit messages or real-time I/O data transfers.

connection

A virtual circuit between two or more network devices, created prior to the transmission of data. After a connection is established, a series of data is transmitted over the same communication path, without the need to include routing information—including source and destination address—with each piece of data.

connection originator

The EtherNet/IP network node that initiates a connection request for I/O data transfer or explicit messaging.

connectionless

Describes communication between two network devices, whereby data is sent without prior arrangement between the two devices. Each piece of transmitted data also includes routing information—including source and destination address.

consumer

See producer/consumer, below.

CSMA/CD

(*carrier sense multiple access with collision detection*) An Ethernet and IEEE 802.3 media access method, operating at the physical layer and handled fully by hardware at the communication media signal level. All network devices contend equally for access to transmit. When a device (device ‘A’) detects a signal sent by another device (device ‘B’) while A is transmitting, A aborts its transmission and retries after a random period of time.

D

DHCP

(*dynamic host configuration protocol*) An extension of the BOOTP communications protocol that provides for the automatic assignment of IP addressing settings—including IP address, subnet mask, gateway IP address, and DNS server names. DHCP does not require the maintenance of a table identifying each network device. The client identifies itself to the DHCP server using either its MAC address, or a uniquely assigned device identifier. The DHCP service utilizes UDP ports 67 and 68.

DNS

(*domain name server/service*) A service that translates an alpha-numeric domain name into an IP address, the unique identifier of a device on the network.

domain name

An alpha-numeric string that identifies a device on the internet, and which appears as the primary component of a web site’s Uniform Resource Locator (URL). For example, the domain name "schneider-electric.com" is the primary component of the URL "www.schneider-electric.com".

Each domain name is assigned as part of the Domain Name System, and is associated with an IP address.

Also called a host name.

DTM

(*device type manager*) A DTM is a device driver running on the host PC. It provides a unified structure for accessing device parameters, configuring and operating the devices, and diagnosing events. DTMs can range from a simple Graphical User Interface for setting device parameters to a highly sophisticated application capable of performing complex real-time calculations for diagnosis and maintenance purposes. In the context of a DTM, a device can be a communications module or a remote device on the network.

E

EDS

(*electronic data sheet*) EDS are simple text files that describe the configuration capabilities of a device. EDS files are generated and maintained by the manufacturer of the device.

Ethernet

A 10 or 100 Mb/s, CSMA/CD, frame-based LAN that can run over twisted pair or fiber optic cable, or wireless. The IEEE standard 802.3 defines the rules for configuring a wired Ethernet network; the IEEE standard 802.11 defines the rules for configuring a wireless Ethernet network.

EtherNet/IP™

A network communication protocol for industrial automation applications that combines the standard internet transmission protocols of TCP/IP and UDP with the application layer Common Industrial Protocol (CIP) to support both high speed data exchange and industrial control. EtherNet/IP employs electronic data sheets (EDS) to classify each network device and its functionality. Because EtherNet/IP is based on standard Ethernet protocols, it can be implemented using commercially available Ethernet components and cabling.

Explicit Messaging

TCP/IP-based messaging for Modbus TCP and EtherNet/IP. It is used for point-to-point, client/server messages that include both data—typically unscheduled information between a client and a server—and routing information. In EtherNet/IP, Explicit Messaging is considered Class 3 type messaging, and can be connection-based or connectionless.

Explicit Messaging client

Explicit Messaging client class) Device class defined by the ODVA for EtherNet/IP nodes that only support Explicit Messaging as a client. HMI and SCADA systems are the most common examples of this device class.

F

full duplex

The ability of a two networked devices to independently and simultaneously communicate with each other in both directions.

G

gateway

A device that interconnects two different networks—sometimes with different network protocols. When used to connect networks based on different protocols, a gateway converts a datagram from one protocol stack into the other. When used to connect two IP-based networks, a gateway (also called a router) has two separate IP addresses - one on each network.

H

hub

A multiport device used to span longer network distances by connecting several Ethernet devices with shielded/unshielded twisted pair or fiber optic cables. Messages received by a hub are repeated on all ports. All connected devices are part of the same segment, share bandwidth and operate via half-duplex communication. A hub lacks the ability to filter network messages based on their source and destination address. Because communication is half-duplex, the likelihood of collisions is increased. Collisions are handled by each connected device using CSMA/CD. Hubs are OSI Layer 1 (physical layer) devices.

I

Implicit Messaging

UDP/IP-based class 1 connected messaging for EtherNet/IP. Implicit messaging maintains an open connection for the scheduled transfer of control data between a producer and consumer. Because an open connection is maintained, each message contains primarily data—without the overhead of object information—plus a connection identifier.

IP address

The 32-bit identifier—consisting of both a network address and a host address—assigned to a device connected to a TCP/IP network.

L

local slave

Functionality offered by Schneider Electric EtherNet/IP communication modules that allows a Scanner to take the role of an Adapter. The local slave enables the module to publish data via Implicit Messaging connections. Local slave is typically used in peer-to-peer exchanges between PLCs.

M

multicast

A special form of broadcast where copies of the packet are delivered to only a subset of all possible destinations. Implicit Messaging typically uses multicast format for communications in an EtherNet/IP network.

O

O->T

Originator to target.

originator

In EtherNet/IP a device is considered the originator when it initiates a CIP connection for Implicit or Explicit Messaging communications; or when it initiates a message request for un-connected Explicit Messaging.

P

producer/consumer

CIP, the core protocol for EtherNet/IP, uses the producer/consumer model, as opposed to the client/server message addressing scheme employed by Modbus TCP. The producer/consumer model is inherently multicast. Nodes on the network determine if they should consume the data in a message based on the connection ID in the packet.

Q

QoS

Quality of Service The practice of assigning different priorities to traffic types for the purpose of regulating data flow on the network. In an Industrial network, QoS can help provide a predictable level of network performance.

R

rack optimized connection

Data from multiple I/O modules are consolidated in a single data packet to be presented to the Scanner in an Implicit Message in an EtherNet/IP network.

RPI

(requested packet interval) The time period between cyclic data transmissions requested by the Scanner. EtherNet/IP devices will publish data at the rate specified by the RPI assigned to them by the Scanner. Modbus TCP devices will receive message requests from the Scanner at each RPI.

S

scanner

A Scanner acts as the originator of I/O connection requests for Implicit Messaging in EtherNet/IP, and message requests for Modbus TCP.

Scanner Class device

A Scanner Class device is defined by the ODVA as an EtherNet/IP node capable of originating exchanges of I/O with other nodes in the network.

subnet mask

The 32-bit value used to hide (or mask) the network portion of the IP address and thereby reveal the host address of a device on a network using the IP protocol.

switch

A multiport device used to segment the network and limit the likelihood of collisions. Packets are filtered or forwarded based upon their source and destination addresses. Switches are capable of full-duplex operation and provide full network bandwidth to each port. A switch can have different input/output speeds (for example, 10, 100 or 1000Mbps). Switches are considered OSI layer 2 (data link layer) devices.

T

T->O

Target to originator.

target

In EtherNet/IP a device is considered the target when it is the recipient of a connection request for Implicit or Explicit Messaging communications; or when it is the recipient of a message request for un-connected Explicit Messaging.

TCP

(*transmission control protocol*) TCP is the OSI transport layer protocol that supports connection-oriented communications, by establishing the connection necessary to transmit an ordered sequence of data over the same communication path.

TCP/IP

Also known as "Internet protocol suite", TCP/IP is a collection of protocols used to conduct transactions on a network. The suite takes its name from the two most broadly used protocols: transmission control protocol and internet protocol. TCP/IP is a connection-oriented protocol that is used by Modbus TCP and EtherNet/IP for Explicit Messaging.

trap

A trap is an event directed by an SNMP agent that indicates either:

- a change has occurred in the status of an agent, or
- an unauthorized SNMP manager device has attempted to get data from, or change data on, an SNMP agent

U

UDP

(*user datagram protocol*) UDP is a transport layer protocol that supports connectionless communications. Applications running on networked nodes can use UDP to send datagrams to one another. Unlike TCP, UDP does not attempt to provide deterministic delivery or ordering of datagrams. However, by avoiding the overhead required by deterministic delivery and checking of datagrams, UDP is faster than TCP. UDP may be the preferred protocol for time-sensitive applications, where dropped datagrams are preferable to delayed datagrams. UDP is the primary transport for Implicit Messaging in EtherNet/IP.

unconnected messaging

In EtherNet/IP, unconnected messaging uses TCP (without a CIP connection) to send explicit messages. More overhead is contained within each unconnected message than for a connected message. The unconnected message is not necessarily provided destination node resources. Unconnected Messaging is used for non-periodic requests.

Index

L

M

N

O

P

R

S

T

U

V